UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

PORTSERVICE-BR: UMA PLATAFORMA PARA PROCESSAMENTO DE LINGUAGEM NATURAL PARA LÍNGUA PORTUGUESA

WANDERSON ROCHA DE ALMEIDA

VITÓRIA
2017

WANDERSON ROCHA DE ALMEIDA

PORTSERVICE-BR: UMA PLATAFORMA PARA PROCESSAMENTO DE LINGUAGEM NATURAL PARA LÍNGUA PORTUGUESA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática. Programa de Pós-Graduação em Informática. Universidade Federal do Espírito Santo. Orientador: Prof. Dr. Orivaldo de Lira Tavares.

VITÓRIA
2017

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)
Sandra Mara Borges Campos – CRB-6 ES-000593/O

A447p

Almeida, Wanderson Rocha de, 1984-
PortService-BR : uma plataforma para processamento de linguagem natural para a língua portuguesa / Wanderson Rocha de Almeida. – 2017.
123 f. : il.

Orientador: Orivaldo de Lira Tavares.
Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Processamento de linguagem natural. 2. Serviços da Web. 3. Língua brasileira de sinais – Aprendizagem. 4. Língua portuguesa – Aprendizagem. 5. Tradução automática. 6. Arquiteturas pedagógicas. I. Tavares, Orivaldo de Lira. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

PortService-Br: Uma Plataforma para Processamento de Linguagem Natural para a Língua Portuguesa

Wanderson Rocha de Almeida

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 18 de dezembro de 2017:

Prof. Dr. Orivaldo de Lira Tavares
Orientador

Prof. Dr. Davidson Cury
Membro Interno

P/ Prof. Dr. Luis Cláudius Coradine
Membro Externo

P/ Prof. Dr. Patrick Henrique da Silva Brito
Membro Externo

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 18 de dezembro de 2017.

Dedico esta dissertação para todos aqueles que de alguma forma direta ou indiretamente contribuíram para que fosse possível sua conclusão, em especial minha mãe, esposa, filhos e irmãos. Dedico também ao meu orientador, pela paciência e dedicação com minha orientação.

AGRADECIMENTOS

A minha mãe querida Euzi Rocha de Almeida, que me criou com tanto carinho e atenção e fez de mim o homem que hoje sou.

A minha esposa, que diante de toda a minha caminhada sempre me deu apoio e carinho nas horas mais difíceis.

Ao meu orientador Prof. Orivaldo de Lira Tavares cujas ideias e orientações sempre me levaram para o melhor caminho, seus ensinamentos permaneceram sempre comigo onde quer que eu esteja. Um profissional de extrema competência, de muita paciência e acima de tudo um grande amigo, que guardarei por toda minha vida.

Aos professores do Programa de Pós-Graduação em Informática (PPGI) e do Laboratório de Informática na Educação (LIEd), da Universidade Federal do Espírito Santo (UFES): ao Prof. Davidson Cury (Dedê) que com seu bom humor sempre tinha uma palavra de incentivo; ao Prof. Crediné Menezes (Índio Velho) que mesmo a distância me passou valiosos conhecimentos principalmente sobre o meio acadêmico. Aos amigos do LIEd, pela paciência e cooperação na troca de ideias e conhecimentos e claro, pela amizade que iremos levar adiante.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela oportunidade de estudar com boas condições para dar seguimento a minha pesquisa.

Principalmente a Deus, pois sem ele não somos nada. Pelos amigos e por tudo.

"Não se mede o valor de um homem pelas suas roupas ou pelos bens que possui, o verdadeiro valor do homem é o seu caráter, suas ideias e a nobreza dos seus ideais."

Charles Chaplin

RESUMO

As ferramentas de Processamento de Linguagem Natural atuais, geralmente, funcionam em um tipo de sistema operacional específico e, em alguns casos, em versões especiais desses sistemas. Essa restrição de acesso a serviços de PLN gratuitos e de fácil acesso, cria dificuldades nas atividades de pesquisa e desenvolvimento dessa área. Esta dissertação foi desenvolvida para diminuir essas dificuldades, por meio do desenvolvimento de serviços de uma API REST em uma plataforma que oferece livre acesso a programadores, pesquisadores e demais interessados. São oferecidos serviços de segmentação de sentenças e palavras, etiquetagem morfossintática usando os etiquetadores TreeTagger, Freeling e um etiquetador desenvolvido no âmbito desta dissertação. Também são oferecidos recursos de análise léxico semântica, com auxílio das WordNets da língua portuguesa ONTO.PT, PULO e OpenWordNet-PT. Os serviços implementados foram testados, em três casos práticos, com diferentes clientes e plataformas, tais como, uma API de tradução de Português para Libras, na identificação de implicações significantes de Piaget, com suporte computacional e como recursos digitais da plataforma CAP-APL - Construtor de Arquiteturas Pedagógicas para a Aprendizagem de Português e de LIBRAS.

Palavra-chave: *Processamento de Linguagem Natural, API REST, Serviços Web, Arquitetura Pedagógica, Aprendizagem de LIBRAS, Aprendizagem de Português, Tradução automática.*

ABSTRACT

Current tools for Natural Language Processing generally work on a particular operating system type and, in some cases, on special versions of those systems. This restriction of access to free and easily accessible PLN services creates difficulties in the research and development of this area. This dissertation was developed to reduce these difficulties by developing services of a API REST in a platform that offers free access to programmers, researchers and other interested people. The following services are offered: segmentation of sentences and words, morphosyntactic labeling using TreeTagger, Freeling and a tagger developed by the author of this dissertation. Semantic lexical analysis resources are also offered, with the help of the following WordNets of the Portuguese language: ONTO.PT, PULO and OpenWordNet-PT. The implemented services were tested in three practical cases with different clients and platforms, such as a translation API of Portuguese for LIBRAS, in the identification of significant implications *of* Piaget with computational support and *as digital* resources of the platform CAP-APL - Constructor of Pedagogical Architectures for the Learning of Portuguese and of LIBRAS.

Keyword: *Natural Language Processing, API REST, Web Services, Pedagogical Architecture, LIBRAS learning, Portuguese learning, Machine translation.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS

LC	Linguística Computacional
PLN	Processamento de Linguagem Natural
LCO	Linguística de Corpus
TA	Tradução Automática
RI	Recuperação da Informação
WN	WordNet
UWM	Universal WordNet
LIBRAS	Língua Brasileira de Sinais
MT	Memória de Tradução
UT	Unidade de Tradução
AP	Arquitetura Pedagógica
CAP-APL	Construtor de Arquiteturas Pedagógicas para a Aprendizagem de Português e de LIBRAS.
REM	Reconhecimento de Entidades Nomeadas

SUMÁRIO

1 INTRODUÇÃO

Como qualquer outro animal, os primeiros seres humanos passaram a ter necessidade de comunicar e interagir entre si. Apesar de ser difícil descrever com exatidão o surgimento de algum tipo de forma de linguagem, estudos apontam para que as primeiras linguagens tenham aparecido há cerca de dois milhões de anos, com os nossos ancestrais (Vieira, 2015).

Comunicação é a transmissão de uma mensagem ou uma informação. Para tanto são usados métodos perceptíveis por ambas as partes. O uso de linguagens engloba, então, a comunicação pela fala, pela escrita, por gestos ou por toque e, por mais diferentes que pareçam, todas possuem aspectos comuns.

À medida em que o ser humano foi evoluindo, a comunicação começou a ser cada vez mais necessária, dessa forma originando a linguagem natural. Com o passar dos tempos, a crescente evolução tecnológica e a grande importância da comunicação no cotidiano dos seres humanos, começaram a despontar campos de estudo visando investigar como tirar proveito das tecnologias atualmente existentes para ajudar o ser humano a melhor aproveitá-las, no que diz respeito ao processamento de uma quantidade cada vez maior de informação, principalmente textual.

Com a disseminação de máquinas capazes de processar dados e dada a importância da linguagem, surgiram ramos da ciência e da engenharia que pretendem tirar partido da automação para que o ser humano possa processar, de forma mais eficiente, os vários suportes em que se usam linguagens. É neste contexto que surge a área de Linguística Computacional (LC) (Jurafsky e Martin, 2014).

A LC é uma área multidisciplinar que lança mão tanto de técnicas de informática quanto de linguística. O uso conjunto dessas técnicas possibilita a construção de sistemas capazes de reconhecer padrões sobre entradas em linguagem natural e produzir informações sobre elas, entre outras aplicações. A LC pode ser dividida em duas categorias: Linguística de Corpus (LCO) e Processamento de Linguagem Natural (PLN). Os esforços da LCO são aplicados ao estudo da língua pela observação de corpus eletrônicos. Já o PLN tem foco na construção de ferramentas

computacionais para estudo da linguagem (Domingues, Favero e Medeiros, 2008).

1.1 Apresentação do problema

Com o surgimento de novas mídias e a popularização das redes sociais, a quantidade de informação produzida a cada segundo vem se tornando cada vez maior e, por conta disso, o uso de PLN vem sendo cada vez mais comum em diversos contextos (Thiele, 2015).

Embora grande parte dessas informações se encontre em formato multimídia, uma enorme quantidade de informação se encontra em formato de texto, fazendo com que as abordagens baseadas em PLN sejam fortes candidatas para basear a construção de soluções que acompanhem esta crescente demanda de processamento de informações textuais, com um nível satisfatório de acurácia e desempenho (Thiele, 2015).

Muitas aplicações de PLN somente funcionam em um tipo de sistema operacional específico, muitas vezes, em apenas algumas versões desses sistemas. Além disso, os ambientes para o funcionamento dessas ferramentas são de difícil configuração e dependem de muitas condições para funcionarem plenamente. Esses fatos dificultam o uso dessas ferramentas de PLN. Muitas dessas aplicações também são de uso comercial, somente podendo ser acessadas mediante prévio cadastro, normalmente com alto custo de utilização.

1.2 Motivação

Sistemas de PLN exigem serviços básicos já resolvidos, mas que estão disponíveis de modo inadequado, quer seja por não serem de livre acesso ou por exigirem esforço e conhecimento técnico de programação para serem instalados e usados. Essas dificuldades passam também pela falta de documentação e pelos projetos inadequados das interfaces das ferramentas.

Devido a essas dificuldades, é comum o pesquisador optar por implementar a sua própria ferramenta para resolver o problema, apesar de haver soluções já implementadas. Sendo assim, o desenvolvimento da presente dissertação contribui para viabilizar e facilitar o acesso e uso dessas ferramentas de PLN.

1.3 Objetivo

Esta dissertação apresenta o desenvolvimento de uma plataforma Web, que visa disponibilizar ferramentas e recursos de PLN de forma gratuita. Dentre elas estão os tokenizadores de sentenças e palavras, etiquetadores morfossintáticos e recursos de análise léxico-semântica.

Essa plataforma pode ser acessada por qualquer aplicativo, a partir de um computador remoto ou dispositivo móvel. Dessa forma, as aplicações clientes dispõem de uma maneira simples de acessar funcionalidades de PLN, fáceis de serem instaladas e configuradas para serem usadas diretamente na plataforma Web ou em aplicações desenvolvidas em qualquer linguagem de programação.

Para a tarefa de etiquetagem morfossintática são usados os etiquetadores TreeTagger (Schmid, 2013) e Freeling (Padró e Stanilovsky, 2012). Também é usado um etiquetador especialmente desenvolvido nesta pesquisa, a ser apresentado em seções posteriores. Como recursos léxico-semânticos são usados o OpenWordNet-PT, o ONTO.PT e o PULO Wordnets (WN) da língua portuguesa.

A plataforma oferecida pode ser consultada por essas aplicações remotamente, livrando as mesmas da tarefa de manter e atualizar repositórios de conhecimento linguístico próprios. Dessa forma evita redundância, por manter os dados em apenas um repositório, e facilita a atualização do conteúdo, com repercussão imediata em todas as aplicações cliente.

Essas funcionalidades podem auxiliar em várias aplicações, tendo em vista que elas usam frequentemente os serviços oferecidos. Entre essas aplicações estão: Tradução Automática (TA), Recuperação de Informação (RI), Sistemas de Perguntas e Respostas, Desambiguação Lexical de Sentidos de Palavras, Corretores Ortográficos e Gramaticais, Análise de Opiniões e Sentimentos, entre outras.

1.4 Questões de investigação

O desenvolvimento de uma plataforma de PLN envolve muitos aspectos importantes, tais como: a disponibilidade, a forma de representação dos dados e os serviços e recursos a serem oferecidos.

QI1 - Existem plataformas de PLN semelhantes à proposta nessa dissertação?

QI2 - Quais recursos e ferramentas podem ser oferecidos pela plataforma?

QI3 - Como usar os serviços oferecidos pela plataforma em aplicações computacionais?

QI4 - Os serviços oferecidos podem ser usados em aplicações voltadas a informática na educação?

QI5 - Os serviços oferecidos podem ser usados em tradutores automáticos de Português-Libras?

1.5 Metodologia

Nesta seção são descritos os métodos usados para o desenvolvimento desta dissertação, tais como levantamento bibliográfico, desenvolvimento do software e validação das ideias apresentadas bem como para a busca das respostas às questões de investigação.

Para responder às QI1 e QI2 foram realizadas inicialmente pesquisas referenciais nos principais portais acadêmicos, como nos portais de periódicos da CAPES, ACM Digital Library e Google Acadêmico.

As palavras-chaves utilizadas foram combinadas de acordo com a ordem estabelecida a seguir: [1] “plataformas”, [2] “processamento de linguagem natural”, [3] “ferramentas de processamento de linguagem natural”. O mesmo conjunto de palavras-chave foi utilizado para realizar pesquisas com as palavras traduzidas para a língua inglesa com o intuito de levantar publicações internacionais sobre o tema.

Foram pré-selecionados cinquenta e cinco títulos entre artigos científicos e dissertações de mestrado. Deste grupo foram selecionados trinta no final da catalogação, leitura e fichamento de todos os artigos. Dos trinta títulos selecionados três apresentavam plataformas de processamento natural, respectivamente para as línguas chinesa, turca e portuguesa.

O levantamento dos requisitos da plataforma foi especificado a partir da leitura e fichamento dos artigos e dissertações levantadas, buscando as principais

dificuldades encontradas por pesquisadores de PLN nos quesitos de instalação, configuração e uso das ferramentas e plataformas disponíveis para o PLN.

Como resultado da pesquisa foi desenvolvida a plataforma PortService-Br, que visa oferecer serviços e recursos de PLN para a língua portuguesa, com APIs específicas para cada tipo de serviço oferecido. A plataforma é de uso gratuito sem necessidade de qualquer tipo de pagamento ou cadastro prévio. A plataforma dispõe de recursos tais como, segmentação de texto em palavras, separação de textos em frases, etiquetagem morfossintática de textos e análise léxico semântica de textos.

Para responder a QI3, QI4, QI5 a plataforma PortService-Br foi testada com aplicações reais, a saber:

- No âmbito do projeto de mestrado de Patrícia Teodoro Gaudio Rios (Rios, 2016). O projeto fez uso da API de etiquetagem morfossintática usando o etiquetador TreeTagger, especificamente na lematização de verbos.
- Uma API de tradução automática de Português para a Língua Brasileira de Sinais (LIBRAS), desenvolvido no âmbito desta dissertação de mestrado como forma de avaliar as funcionalidades da plataforma.
- No âmbito do projeto CAP-APL Plataforma para criação e uso de arquiteturas pedagógicas para aprendizagem de Português e Libras.

A modelagem da plataforma inclui as etapas de especificação de requisitos, de análise e de projeto, apresentadas no quarto e no quinto capítulo desta dissertação.

O desenvolvimento do PortService-Br inclui uma arquitetura MVC (Model/View/Controller), com o uso do *framework* Web Django (Django, 2017), em conjunto com a linguagem de programação Python. Para o desenvolvimento da API de tradução, foi usado ASP.NET Core 2.0 - Microsoft (2017) e .NET Standard 2.0 - Microsoft (2016), em conjunto a linguagem de programação C#.

1.6 Estrutura da dissertação

A estrutura da dissertação foi dividida em 7 capítulos, sendo o primeiro uma introdução, com a definição do problema, motivação, objetivo, questões de investigação e metodologia. O segundo capítulo versa sobre as ferramentas e

plataformas para processamento de linguagem natural, abordando seus principais recursos e analisando plataformas semelhantes à proposta deste trabalho. O terceiro capítulo aborda os conceitos e aplicações das WordNets. O quarto capítulo apresenta a análise e projeto da plataforma, o projeto da arquitetura e modelos de classes, bem como especificação de requisitos da plataforma, seus desafios e objetivos. O quinto capítulo apresenta a implementação da plataforma, bem como apresenta detalhes de uso, parâmetros de requisição e estrutura dos arquivos de resposta a requisições. O sexto capítulo apresenta os resultados e aplicações que fazem uso dos serviços da plataforma. E finalmente no capítulo sete são apresentadas as considerações finais.

2 FERRAMENTAS E PLATAFORMAS DE PROCESSAMENTO DE LINGUAGEM NATURAL

O objetivo deste capítulo é apresentar o levantamento de algumas das ferramentas e recursos disponíveis para o PLN, bem como apresentar alguns serviços e plataformas semelhantes à proposta nesta dissertação.

2.1 Conceitos básicos

Com a crescente evolução do PLN, especialmente nas últimas décadas, houve significativo aumento nos problemas e suas soluções, e neste momento esta área do conhecimento tem aplicações em diversos setores, tais como: economia, medicina, entre outras. No cenário internacional das pesquisas em computação, o PLN faz parte da agenda de pesquisa das mais importantes universidades e centros de pesquisa. O PLN conta atualmente com ferramentas e plataformas estáveis e também complexas, que propõe aplicações para alguns dos problemas comuns desta área de conhecimento.

2.2 O estado da arte

O Objetivo desta dissertação de mestrado é implementar uma plataforma *web*, que visa disponibilizar um conjunto de ferramentas de PLN em uma API REST. Dessa forma torna-se necessário a investigação da evolução das ferramentas de PLN. O estado da arte tem como objetivo a listagem do conhecimento atual nos campos com as ferramentas de PLN usadas nesta dissertação.

2.2.1 Segmentação de texto em frases

A segmentação de um texto em frases, em um primeiro momento pode parecer uma tarefa simples, porém levanta vários problemas inerentes à língua e que devem ser abordados de forma rigorosa.

O processamento de textos geralmente pressupõe a habilidade de dividi-lo em sentenças individuais. A atividade de Segmentação de Texto, é justamente o mecanismo que provê a segmentação do conteúdo de texto plano em sentenças. Cada sentença limita um conjunto semântico mínimo para definição de uma proposição (De Lima et al., 2007).

A segmentação de textos é aplicada por meio da identificação de caracteres finalizadores de sentenças, principalmente, dos sinais de pontuação. Embora se use o ponto final (ou ponto de interrogação ou exclamação) na maioria das línguas para indicar o fim de frase, nem sempre isso acontece. A dificuldade desta tarefa está em associar corretamente os sinais de pontuação ao fim de sentença, uma vez que tais sinais também podem demarcar abreviações de nomes, separação de dígitos em datas, horas, números de telefones e números ordinais (Vieira et al., 2015).

2.2.2 Segmentação de texto em palavras

A segmentação de um texto equivale a sua segmentação em palavras ou identificação de *tokens*, ou seja, elementos de cada oração, que podem ser palavras, números e sinais de pontuação. Essa tarefa utiliza, basicamente os sinais gráficos tais como espaços e algoritmos para o reconhecimento de entidades limítrofes de um *token*. Contudo, além de existirem línguas onde o uso do espaço não é semelhante às línguas mais convencionais como no caso das línguas orientais, também há determinados caracteres, como o hífen ou a apóstrofe, que podem funcionar como separadores ou como conectores entre duas palavras (De Lima et al., 2007) e (Vieira et al., 2015).

A utilização de espaços em branco, de maneira geral, nem sempre pode definir da forma mais adequada as fronteiras das palavras, principalmente em palavras compostas e nomes próprios. Por isso, a recomendação é o uso conjunto de tokenizadores e técnicas de reconhecimento de entidades nomeadas (De Lima et al., 2007).

2.2.3 Lematização e radicalização

O processo de radicalização tem como tarefa principal a remoção de sufixos e prefixos de um termo, para que este seja reduzido ao seu radical (*stem*). O radical de uma palavra é a parte comum a todas as palavras derivadas, ou seja, da mesma palavra raiz, exemplo: “prender”, “desprender”, “desprendido”. As três palavras têm como radical (prend) (Neves, 2010).

A raiz pode até nem ser uma palavra da língua em questão, como no exemplo acima. Algoritmos de radicalização tem grande utilidade pela facilidade com que se implementa, sem necessidade de um léxico completo. Essa funcionalidade pode

contribuir de forma expressiva na recuperação de informação em cerca de 35% na eficácia (Vieira et al., 2015).

Algoritmos de radicalização para línguas ocidentais possuem oito tarefas distintas: remover o plural, passar a palavra para o masculino, retirar conteúdo adverbial, retirar diminutivos e aumentativos, retirar sufixos nominais, verbais ou vogais terminais e, por fim, retirar a acentuação. Nem sempre esta técnica é eficaz, pois, as heurísticas usadas não funcionam bem para todas as palavras. Quando palavras distintas podem ser analisadas como tendo a mesma raiz, esse fato aumenta a ambiguidade.

A técnica de lematização em um determinado texto consiste em representar cada palavra na sua forma primitiva (*lemma*). A técnica auxilia o processamento do texto reduzindo a sua complexidade, podendo reduzir um conjunto de palavras da mesma família apenas ao seu lema. Por exemplo, para lematizar substantivos é usada sua forma masculina singular, para lematizar verbos é usado o infinitivo. Por exemplo, após ser aplicado um processo de lematização às palavras “gostaríamos”, “gostei”, “gosto” e “gostaste”, estas resultam na forma primitiva “gostar”. Embora também exista ambiguidade neste processo, ainda assim é mais informativo que a radicalização (De Lima et al., 2007) e (Vieira et al., 2015).

2.2.4 Etiquetagem morfossintática (Part-Of-Speech (POS) Tagging)

A etiquetagem morfossintática é uma tarefa básica requerida por inúmeras aplicações de PLN, tais como análise gramatical, tradução automática e processamento de fala (Domingues, Favero e Medeiros, 2008).

A tarefa de *Part-Of-Speech* (POS) *tagging*, palavras de um texto são selecionadas e classificadas, de acordo com sua distribuição sintática e morfológica e a cada uma é atribuída uma *tag*, que corresponde a essa classificação. Nessa abordagem, o analisador recebe uma cadeia de itens lexicais e um conjunto específico de etiquetas como entrada, e produz um conjunto de itens lexicais com a melhor etiqueta associada a cada item. Basicamente existem 3 tipos e técnicas de POS *tagging* citados na literatura (Domingues, Favero e Medeiros, 2008):

- Baseados em regras, caracterizada pelo uso de regras de etiquetagem

codificadas manualmente por linguistas.

- Probabilística, que usa métodos de etiquetagem estatística. Nesse tipo de etiquetagem, cada palavra possui um conjunto finito de etiquetas possíveis. A ideia é rotular palavras com suas etiquetas mais prováveis, tendo como referência um corpus de treinamento etiquetado.

- Híbrida, que envolve a combinação das técnicas baseadas em regras e probabilística.

2.2.5 Freeling

O *Freeling* é uma biblioteca de livre utilização para análise linguística, desenvolvida por Luís Padró (Padró e Stanilovsky, 2012), no TALP Reseach Center. Inclui recursos linguísticos para 13 línguas diferentes, dentre elas o Português. Dentre as funcionalidades oferecidas podem ser destacadas:

- atomização (tokenização) e segmentação (divisão em frases) de texto;

- análise morfológica;

- reconhecimento de termos multipalavra;

- etiquetagem morfológica e desambiguação;

- análise de grafo de dependência com base em regras;

- detecção e classificação de entidades nomeadas;

- resolução de correferência nominal;

Apesar do grande número de funcionalidades úteis que o *FreeLing* oferece, o processo de instalação dele é bastante complexo sendo o mesmo composto de muitas dependências para seu pleno funcionamento tendo o mesmo maior eficiência no sistema operacional Linux. Os desempenhos do *FreeLing*, durante os procedimentos de POS *tagging*, são próximos do estado-da-arte, variando entre 94% e 98% de precisão (Padró e Stanilovsky, 2012).

2.2.6 TreeTagger

O TreeTagger foi desenvolvido no Institute for Computational Linguistics da University of Stuttgart. Trata-se de uma ferramenta de uso livre. O método de etiquetagem dele é baseado em árvores de decisão e o seu desempenho em pesquisas e experimentos atingiu 96% de precisão para o Inglês e o Alemão. Para o Português, também foi desenvolvido um *splitter* que separa as contrações e os pronomes clíticos, para além de um reconhecedor de entidades mencionadas que identifica nomes próprios (Schmid,2013). Apesar de ser um ótimo etiquetador, seu processo de instalação é bastante complexo sendo o mesmo composto de muitas dependências para seu pleno funcionamento tendo o mesmo maior eficiência no sistema operacional Linux.

2.2.7 Comparativo de desempenho entre *Freeling* e *TreeTagger*

Para conhecer o desempenho dos etiquetadores *Freeling* e *TreeTagger*, Gamallo e Garcia (2013) realizaram avaliações das ferramentas usando dois corpora da língua portuguesa a saber:

- Bosque_CF
- Miscelâneo

O primeiro foi adaptado Bosque_CF 8.0, criado a partir de CETENFolha, contendo cerca de 81.000 tokens revisados manualmente por linguistas. Este corpus, compilado a partir de artigos do jornal a Folha de São Paulo, pertence à variedade do Português do Brasil. O segundo corpus de teste, Miscelâneo, é um corpus modesto de 600 tokens anotado manualmente especificamente para a avaliação entre o TreeTagger e o Freeling. Foi compilado a partir de textos da Wikipédia e de excertos de obras literárias portuguesas. Todos os textos deste corpus têm como variedade do Português europeu (Gamallo e Garcia, 2013). A Tabela 1 apresenta o resultado das avaliações com Bosque_CF.

Tabela 1.Precisão de Freeling e Treetagger no corpus de teste Bosque_CF.

Etiquetador	Tokens avaliáveis	Acurácia (%)
Freeling	66.612	93,037
TreeTagger	69.118	91,390

A Tabela 2 apresenta os resultados obtidos com o corpus de treinamento Miscelâneo. Neste experimento os resultados se encontram próximos ao estado da arte, situado em cerca de 97%.

Tabela 2. Precisão de Freeling e TreeTagger no corpus de teste Miscelâneo.

Etiquetador	Tokens avaliáveis	Acurácia (%)
Freeling	532	98,308
TreeTagger	529	96,030

Os dois testes mostram que o sistema FreeLing tem um desempenho superior ao de TreeTagger, cujos valores de precisão se situam 2 pontos por baixo dos do primeiro etiquetador. Os resultados indicam que o módulo de PoS-tagging de FreeLing (que utiliza o algoritmo (HMM) atinge melhores resultados do que TreeTagger (baseado em árvores de decisão), com diferenças de 2%.

2.2.8 Tradução automática

A TA é uma das aplicações mais antigas da computação e que vem se tornando cada vez mais necessária diante do mundo globalizado, onde as diferenças linguísticas ainda representam uma barreira para a comunicação e o compartilhamento de informações. Este é talvez o exemplo paradigmático do PLN. O desafio é simples, porém suas soluções nem tanto. Pretende-se a tradução de um texto, de uma determinada língua, a língua de origem, para uma outra, a língua de destino ou língua alvo (Breda, 2008) e (Vieira et al., 2015).

Os sistemas de tradução assistida por computador podem simplesmente oferecer o acesso a dicionários e enciclopédias em tempo real e recursos de processamento de textos, ou podem realizar a verificação ortográfica e gramatical, ou mesmo realizar parte da análise textual. Convém, no entanto, chamar à atenção que este processo obriga a um conjunto de tarefas nada simples: por um lado compreender o texto a traduzir, saber como representar essa semântica computacionalmente, e conseguir gerar, de novo, o texto traduzido, usando uma estrutura sintática semelhante à original.

2.2.9 Análise de sentimento

Análise de Sentimentos ou Mineração de Opinião visa descobrir de forma computacional opiniões e seus conceitos relacionados, como: sentimentos, atitudes, avaliações e emoções relacionadas a alguma entidade, tais como: produtos, serviços, organizações, indivíduos, eventos, tópicos e seus atributos. Tais opiniões podem ser na maioria das vezes positivas ou negativas, e em alguns casos neutras (F Filho et al., 2015). A análise de sentimento é uma das ferramentas com popularidade alta na área de PLN. Para se compreender a sua importância basta perguntar como é que determinada empresa pode recorrer a redes sociais, como o Twitter, para saber a opinião dos seus consumidores (Vieira et al., 2015).

2.2.10 Sumarização

O objetivo da sumarização é reconhecer, em um texto, o que é relevante e o que pode ser descartado, para compor um sumário. Esse é um ponto problemático e sujeito a análise minuciosa por parte dos pesquisadores. A importância de uma sentença ou parte de um texto pode depender de muitos fatores: a) os objetivos do autor do sumário, b) os objetivos ou interesses de seus possíveis leitores e c) da importância relativa e muitas vezes subjetiva que o autor atribui às informações textuais, desta forma fazendo com que mesmo a estruturação do sumário esteja sujeita à complexidade de se determinar forma e conteúdo, a partir da fonte correspondente. Devido a sua utilidade e frequência e aos avanços na área de PLN, é grande o interesse em automatizar o processo de sumarização.

A sumarização automática é objeto de estudo desde os primórdios da computação. No final da década de 1950 surgiram alguns métodos estatísticos com a finalidade

de extrair as sentenças principais de um texto. As pesquisas continuaram avançando nas décadas seguintes, fato esse que trouxe muitos avanços à área, sob a ótica do PLN existem duas abordagens principais: a superficial e a profunda, as quais caracterizam métodos distintos de sumarização automática. A abordagem superficial usa, fortemente, métodos experimentais e estatísticos, enquanto a profunda está relacionada a teorias formais e linguísticas (Martins et al., 2001).

2.3 Plataformas

São encontrados na literatura alguns projetos que disponibilizam plataformas de PLN similares ao descrito neste trabalho. Dentre eles podemos citar três: FudanNLP[1] para a língua chinesa, o ITUNLP[2] para a língua turca e o SplineAPI[3] para a língua portuguesa.

2.3.1 FudanNLP

FudanNLP é um conjunto integrado de aplicações de PLN, para a língua chinesa, de alto desempenho. FudanNLP possui interfaces para análise léxica (segmentação de palavras, etiquetagem morfossintática e reconhecimento de entidades mencionadas) e análise semântica (desambiguação lexical de sentidos e rotulagem de papéis semânticos). Disponibiliza tanto ferramentas quanto *corpus* anotados.

As ferramentas podem ser acessadas de duas formas: remotamente, por meio de consultas a um *web service*, e localmente, por intermédio de bibliotecas de programação (Qiu, Zhang e Huang, 2013). Apesar de ter licença gratuita sua cobrança é baseada no uso e no tempo de processamento de cada requisição ao serviço e os usuários precisam ser previamente cadastrados para usar as APIs do sistema

[1] https://code.google.com/archive/p/fudannlp/

[2] http://tools.nlp.itu.edu.tr/

[3] http://spline.di-um.org/

2.3.2 ITUNLP

ITUNLP é uma plataforma multicamada para língua turca que fornece a pesquisadores e estudantes da área de PLN aplicações de pré-processamento de textos, análise sintática e morfológica e reconhecimento de entidades nomeadas. O ambiente pode ser consultado em três formas: através de uma interface *web* amigável, upload de arquivos e requisições remotas a um *web service* (Eryigit, 2014). É uma ferramenta proprietária de uso comercial.

2.3.3 SplineAPI

SplineAPI é uma plataforma *web* que disponibiliza um conjunto de operações comuns de PLN como: tokenização, etiquetagem morfossintática, lematização, entre outros. A plataforma implementa uma API REST que pode ser acessada remotamente por diversos tipos de aplicações que necessitem de funcionalidades de PLN (Vieira et al., 2015). Trata-se de uma ferramenta privada e sua cobrança é baseada no uso e no tempo de processamento de cada requisição ao serviço e os usuários precisam ser previamente cadastrados para usar as APIs do sistema.

2.4 Considerações finais deste capítulo

Vale ressaltar que nenhuma das três plataformas citadas fornecem o recurso de análise léxico-semântica baseadas em Wordnets proposta nesta dissertação. Também não foram encontradas informações sobre a acurácia das funcionalidades oferecidas.

A versão atual do PortService-BR, disponibiliza serviços de etiquetagem morfossintática, segmentação de textos em frases e sentenças e recursos de análise léxico semântica.

3. WORDNET (WN)

O objetivo deste capítulo é apresentar o modelo de conhecimento proposto em uma WN para representar o conhecimento lexical de uma determinada língua. São apresentados projetos de WNs de livre utilização para a língua portuguesa.

3.1 O modelo wordnet

WNs são bases de conhecimento lexical que formam repositórios de itens lexicais. Dentre as informações contidas nesses repositórios podem ser destacados: os possíveis sentidos das palavras, relações entre sentidos, definições e frases que exemplificam seu uso em determinada língua (Gonçalo Oliveira et al., 2015).

O modelo mais popular deste tipo de recurso é a WN de Princeton ou simplesmente WN.Pr (Fellbaum, 2010). Sua flexibilidade, adaptabilidade para outras línguas e disponibilidade gratuita levou a sua grande aceitação pela comunidade de PLN, tornando-a um padrão (*standard*). WN.Pr foi desenvolvida manualmente por linguistas para o inglês dos EUA, na Universidade de Princeton, e desde então recebe constantes atualizações. Inicialmente baseava-se em princípios psicolinguísticos, com a combinação de informações lexicográficas tradicionais, semelhantes a um dicionário, tendo organização adequada para tratamento computacional, fato esse que facilita seu uso em bases de conhecimento léxico-semânticos (Fellbaum, 2010).

Assim como um tesauro, a WN.Pr é organizada em itens lexicais sinônimos, chamados de synsets, que podem ser vistos como possíveis lexicalizações de um conceito da língua. A WN.Pr também abrange outros tipos de relações semânticas como: hiperonímia, onde um conceito é generalização de outro, meronímia, onde um conceito é parte de outro, entre outras relações. A Figura 1 apresenta um Conjunto reduzido de relações da WN de Princeton para palavra “car”

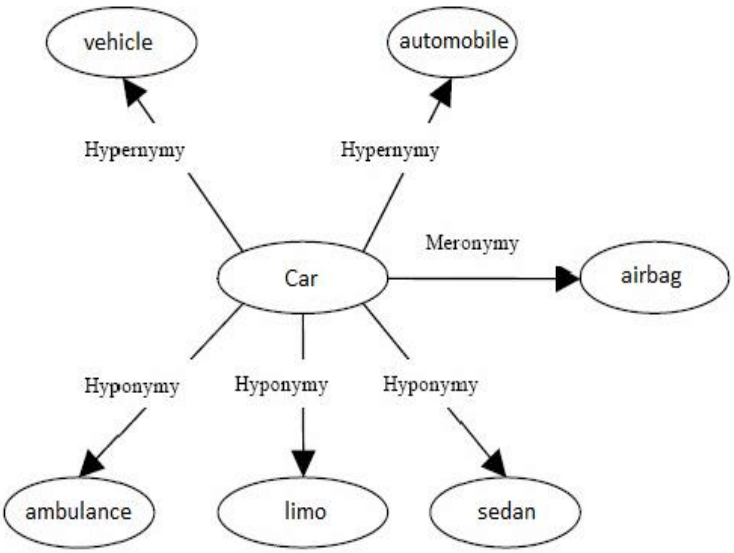

Figura 1. Conjunto reduzido de relações da WN de Princeton para palavra “car”.

Além dessas informações cada synset pertence a uma determinada categoria gramatical (substantivo, verbo, adjetivo ou advérbio), possui uma glosa semelhante a um dicionário com a definição do synset e, opcionalmente, frases que exemplificam seu uso na língua (Fellbaum, 2010). Nesse modelo, uma única palavra pode conter vários sentidos e pode ainda ter diferentes classificações gramaticais dependendo do sentido.

A partir de 2010 surgiram vários projetos de *Wordnets* para a língua portuguesa, com diferentes abordagens de construção e disponibilizados gratuitamente. Dentre os projetos abertos se destacam: PULO (Simões e Guinovart, 2014), Onto.PT (Oliveira e Gomes, 2014) e OpenWordnet-PT (Paiva, Rademaker e Melo, 2012), (Rademaker et al., 2014).

3.2 Onto.PT

Onto.PT é uma ontologia lexical que se encontra em sua versão 0.6. Trata-se de uma WN desenvolvida no âmbito do projeto de doutorado de Hugo Gonçalo Oliveira, no Centro de Informática e Sistemas, da Universidade de Coimbra (Oliveira et al., 2015). O Onto.PT foi desenvolvido de forma automática, aproveitando os recursos desenvolvidos no projeto PAPEL (Gonçalo Oliveira et al., 2008) e no Tep (Maziero et

al., 2008).

A Onto.PT utiliza uma abordagem denominada de ECO para sua construção, diferentemente de abordagens baseadas em tradução, que tentam encontrar correspondências em português, de palavras e synsets de outras línguas. A abordagem ECO tenta aprender de forma automática a estrutura de uma WN, incluindo os conteúdos e limites próprios dos synsets, ou mesmo synsets envolvidos em cada limite da relação (Gonçalo Oliveira e Gomes, 2014).

Em sua versão 0.6 Onto.PT inclui cerca de 169 mil itens lexicais únicos, onde são organizados em cerca de 117 mil synsets, que se relacionam através de 174 mil instâncias de relação. Dessa forma se torna um importante recurso para o PLN da língua portuguesa. A Onto.PT é um pouco diferente das outras WNs, não só pela abordagem de construção, mas também por incluir um vasto conjunto de relações semânticas, também usado no projeto PAPEL. Dessa forma não apenas as relações comuns como sinonímia, hiperonímia e vários tipos de meronímia, mas também relações como causa, finalidade, local e maneira foram adicionadas (Gonçalo Oliveira et al., 2015). A Figura 2 apresenta um conjunto restrito dessas relações para a palavra “estudar”.

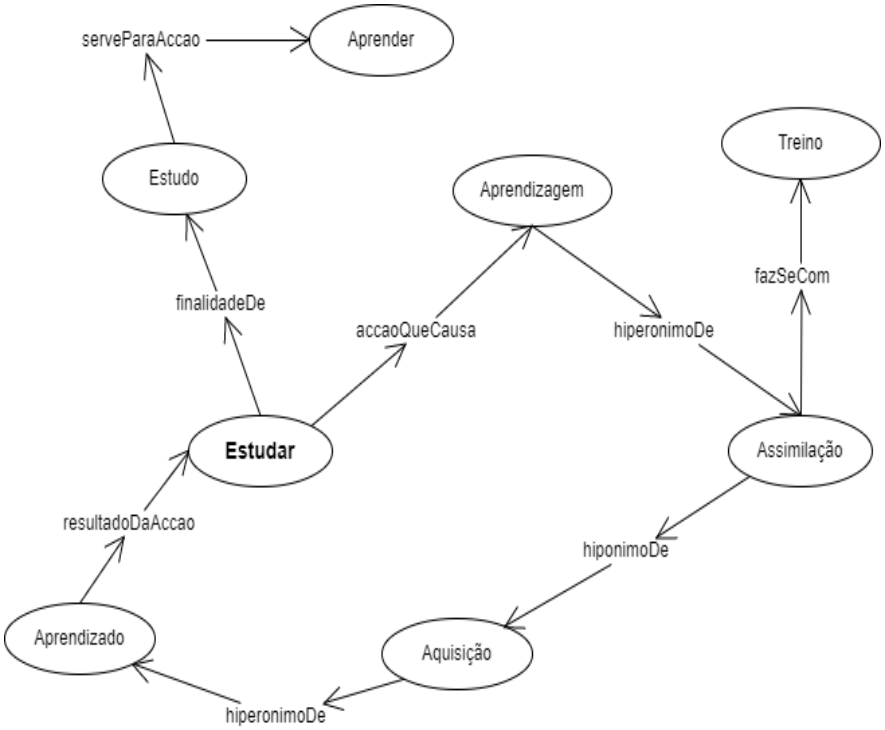

Figura 2. Conjunto restrito de relações do ONTO.PT para a palavra "estudar".

Onto.PT, devido a seu método de construção, é um recurso dinâmico e pode ter variações de uma versão para outra, quanto ao número e abrangência de synsets. Onto.PT é disponibilizado gratuitamente sob forma de arquivos RDF/OWL (Oliveira e Gomes, 2014).

3.2 Portuguese unified lexical ontology (PULO)

A PULO (Simões & Guinovart 2014), abreviatura de Portuguese Unified Lexical Ontology, não deve ser visto como mais uma WN. Pretende, sim, ser o início de um projeto conjunto de disponibilização de uma WN livre para a língua portuguesa.

Esse projeto teve início no final de 2014, consistiu na realização de algumas experiências de alinhamento e tradução entre as versões espanhola, inglesa e galega da WN. Para além dessas mesmas WNs, obtidas do MCR, são usados dicionários probabilísticos de tradução, um dicionário de tradução dinâmico entre as línguas portuguesa e galega, e o vocabulário ortográfico da língua portuguesa (Oliveira et al., 2015). A Figura 3 apresenta um conjunto restrito dessas relações para a palavra “estudar”.

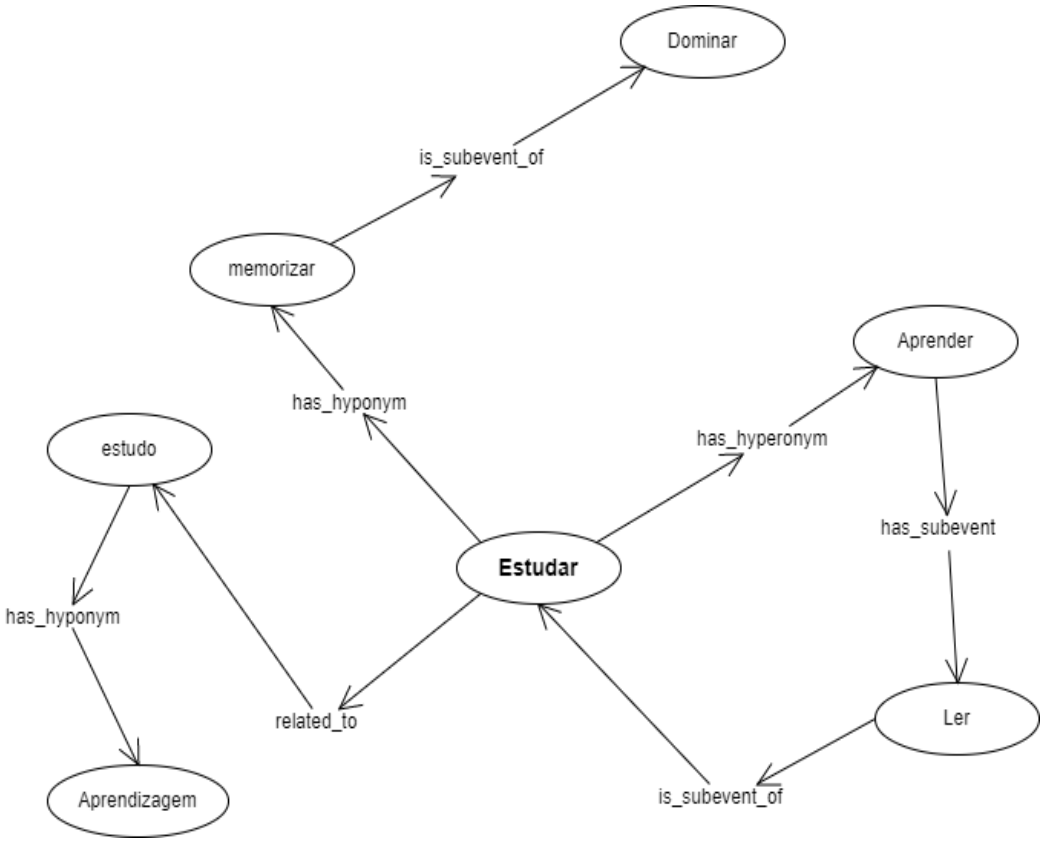

Figura 3. Conjunto restrito de relações do PULO para a palavra "estudar".

A estrutura ontológica é, neste momento, a mesma que a WN.Pr, que é partilhada pelas restantes WNs disponíveis no projeto MCR: inglês, basco, galego, castelhano e catalão. Este processo gerou cerca de 50 mil sentidos de palavras, mas apenas pouco mais de 17 mil foram realmente adicionadas ao PULO. Isto deveu-se à abordagem estatísticas de corte usada. O valor de pontuação, obtido para cada sentido, foi devidamente armazenado na base de dados, de modo a que se possa ter informação da qualidade ou relevância de cada um.

Embora o fato de se usar uma estrutura ontológica semelhante, a estrutura interna da base de dados permite que seja facilmente extensível a novos conceitos (Oliveira et al., 2015). Neste momento, o PULO conta com 17.631 sentidos, referentes a 13.709 synsets diferentes.

3.3 OpenWordNet-PT

A OpenWordNet-PT (De Paiva et al. 2012) (Radmaker et al. 2014) é uma WN desenvolvida originalmente por Valeria de Paiva, Alexandre Rademaker e Gerard de Melo, como uma projeção sintática da Universal WordNet (UWN). A OpenWordNet-PT vem sendo desenvolvida desde 2010 para servir de subsídio léxico para um sistema voltado para raciocínio lógico.

O processo de construção da OpenWN-PT, decorrente do processo de criação da UWN, usa aprendizagem de máquina para construir relações entre grafos representando informações vindas de versões em múltiplas línguas da Wikipédia, bem como de dicionários eletrônicos abertos. Apesar de ter começado como uma projeção apenas ao nível dos lemas em português e suas relações, a OpenWNPT tem sido constantemente melhorada por meio de acréscimos linguisticamente motivados, quer manualmente, quer fazendo uso de grandes corpora, como é o caso do léxico de nominalizações que integra a OpenWN-PT (De Paiva et al. 2014). Uma das características da construção deste último recurso é tentar incorporar os diferentes materiais (de qualidade) já produzidos e disponibilizados para a língua portuguesa.

A OpenWN-PT integra três estratégias linguísticas no seu processo de enriquecimento lexical: (i) tradução; (ii) corpus; (iii) dicionários. Com relação à tradução, são usados léxicos e listas produzidas para outras línguas, como inglês,

francês e espanhol, automaticamente traduzidos e posteriormente revistos. A incorporação de dados de corpus contribui com palavras ou expressões de uso corrente que podem ser específicas da língua portuguesa ou que, por outros motivos, podem não constar nas outras WNs. A Figura 4 apresenta um conjunto restrito de relações do OpenWordNet-PT para a palavra “estudar”.

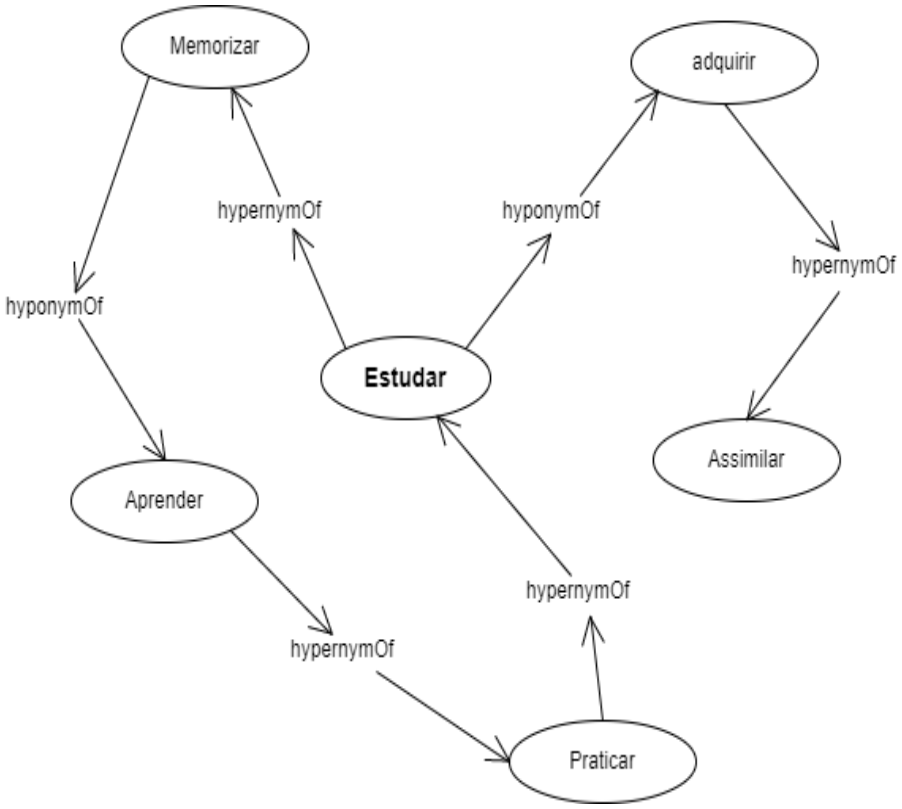

Figura 4. Conjunto restrito de relações do OpenWordNet-PT para a palavra “estudar”.

A OpenWN-PT foi escolhida pelos organizadores dos projetos FreeLing (Padró & Stanilovsky 2012), OMWN (Bond & Foster 2013) e ainda Google Translate como a representante das WNs abertas em português utilizada por esses projetos. Presumivelmente essa escolha se deve à cobertura abrangente da OpenWN-PT e também à sua qualidade. Assim como a Onto.PT, a OpenWN-PT também está disponível em RDF/OWL (Oliveira et al., 2015).

3.4 Comparativos entre ONTO.PT, PULO e OpenWordNet-PT

Após uma descrição das três WNs para o Português, selecionadas para o projeto

PortService-Br, esta seção apresenta um breve comparativo entre as versões mais recentes das WNs. Vale ressaltar que muitos dos indicadores são meramente quantitativos e não consideram a coerência ou a utilidade dos conteúdos. A Tabela 3 apresenta a abordagem seguida para criação e atualização das Wns.

Tabela 3.Abordagem de criação e atualização das WNs.

WordNet	Synsets	Relações	Atualização
Onto.PT	ER+clustering	ER+clustering	automática
OpenWordNet-PT	projeção UWN	transitividade	semi-automática
PULO	triangulação	transitividade	semi-automática

Dentre as abordagens de criação, o PULO destaca-se por utilizar não só a WN.Pr como wordnet base para sua criação, mas também as wordnets de outras línguas como espanhol e galego. Ao contrário de todas as outras, a estrutura da Onto.PT é aprendida de forma completamente automática, com base na extração de relações a partir de outros recursos textuais ou de outras wordnets, e da descoberta de aglomerados (clusters) de sinônimos, que dão origem aos synsets. Já OpenWordNet.PT usa atualizações automáticas e manuais. A Tabela 4 apresenta uma projeção ao nível de itens lexicais abrangidos pela WNs e separadas por categoria gramatical.

Tabela 4. Quantitativo de itens lexicais abrangidos pelas WNS.

WordNet	Substantivos	Verbo	Adjetivo	Advérbio	Total
Onto.PT	97.531	32.958	34.392	3.995	168.876
OpenWordNet-PT	43.996	3.914	5.422	1.388	54.720
PULO	10.260	4032	3.166	173	17.631

Neste quesito Onto.PT se destaca por ter um número bastante expressivo de itens lexicais, chegando a ser três vezes maior que a segunda colocada - OpenWordNet-PT. Isso se deve à sua abordagem de construção, completamente automática. Neste quesito PULO tem baixa cobertura e expressividade. A Tabela 5 apresenta os indicadores de dimensão e cobertura das WNs, número de sentidos de palavras, número de synsets e ainda o número de instâncias de relações e alinhamento com outras WNs.

Tabela 5. Quantitativo de sentidos, synsets e relações nas WNs.

WordNet	Sentidos	Synsets	Relações	Alinhamento
Onto.PT	248.773	117.450	341.506	Nenhum
OpenWordNet-PT	73.802	43.925	74.102	WN.Pr
PULO	17.631	13.709	48.658	MCR

Neste quesito se destaca o Onto.PT com números bastante expressivos. Porém, segundo Oliveira et al (2015), um número muito grande de synsets pode ser sinal de “ruído” no processo de agrupar e/ou no processo de discriminação dos mesmos. É importante perceber a existência de um balanço intrínseco entre o número de synsets e a correção e utilidade das WNs. Um dos grandes gargalos no desenvolvimento de WNs é precisamente decidir, por um lado, se duas palavras que devem ser consideradas como sinônimas e, dessa forma, colocadas em um mesmo synset e, por outro, que palavras têm de estar em synsets diferentes, desafio que desde sempre acompanha a ciência da lexicografia e para o qual, acredita-se, ainda não haja resposta exata.

A versão atual do PortService-Br, disponibiliza recursos das três WNs citadas acima ONTO.PT, OpenWordNet-PT e PULO. Esses recursos são disponibilizados em serviços de uma API REST (serviços web de fácil uso e gratuitos).

4. ANÁLISE E PROJETO DO PORTSERVICE-BR

Este capítulo apresenta a especificação de requisitos da plataforma PortService-Br, bem como a análise e projeto a arquitetura definida para o sistema, sua fundamentação, estrutura interna. Também é apresentada da modelagem dos serviços oferecidos na plataforma.

4.1 Especificação de requisitos

A plataforma tem como objetivo, disponibilizar ferramentas e recursos de PLN para a língua portuguesa. Os serviços oferecidos por essa plataforma são destinados à consultas pessoais diretamente no site da plataforma ou via requisições computacionais HTTP, a uma API REST, feitas por aplicações que fazem uso de PLN. Essas aplicações computacionais podem, por exemplo, ser aplicadas à aprendizagem e à ampliação do conhecimento sobre o léxico da língua portuguesa, ou mesmo, serem usadas em sistemas de tradução que precisam tratar textos em Português.

4.1.1 Problemas e desafios

Ferramentas e recursos de PLN têm sido desenvolvidos por muitos pesquisadores ao longo dos anos, porém grandes são os seus desafios de utilização. Muitas dessas ferramentas e recursos são de difícil instalação e configuração. A grande maioria é desenvolvida para sistemas operacionais específicos e para versões restritas deles. Desse modo, é comum que pesquisadores e aplicações que usem tais ferramentas acabem por desenvolver ferramentas próprias, restritas às respectivas necessidades.

Com base nos problemas e desafios encontrados e após interações com pesquisadores da área de PLN, conclui-se que a melhor estratégia para contornar tais desafios é oferecer tanto ferramentas quanto recursos em forma de serviços Web, mais especificamente como APIs de serviços. Sendo assim, pesquisadores e aplicações com acesso à Internet podem ter acesso aos serviços independente de plataforma, sem a necessidade de instalação e configuração de softwares complexos, bastando apenas fazer a requisição dos serviços remotamente.

4.1.2 Cenários de uso da plataforma PortService-Br

Existem duas formas de uso dos serviços oferecidos pela plataforma PortService-Br. A plataforma pode ser acessada via requisições HTTP para a API REST de cada serviço. Alguns serviços da plataforma também podem ser usados diretamente no site do projeto.

4.1.2.1 Cenários de uso via API REST

Basicamente o uso dos serviços oferecidos no PortService-Br, via API REST, é realizado por aplicações clientes, ou seja, outros sistemas computacionais que usem algum serviço oferecido pela plataforma. A Figura 5 apresenta os casos de uso para esse cenário de consulta à plataforma.

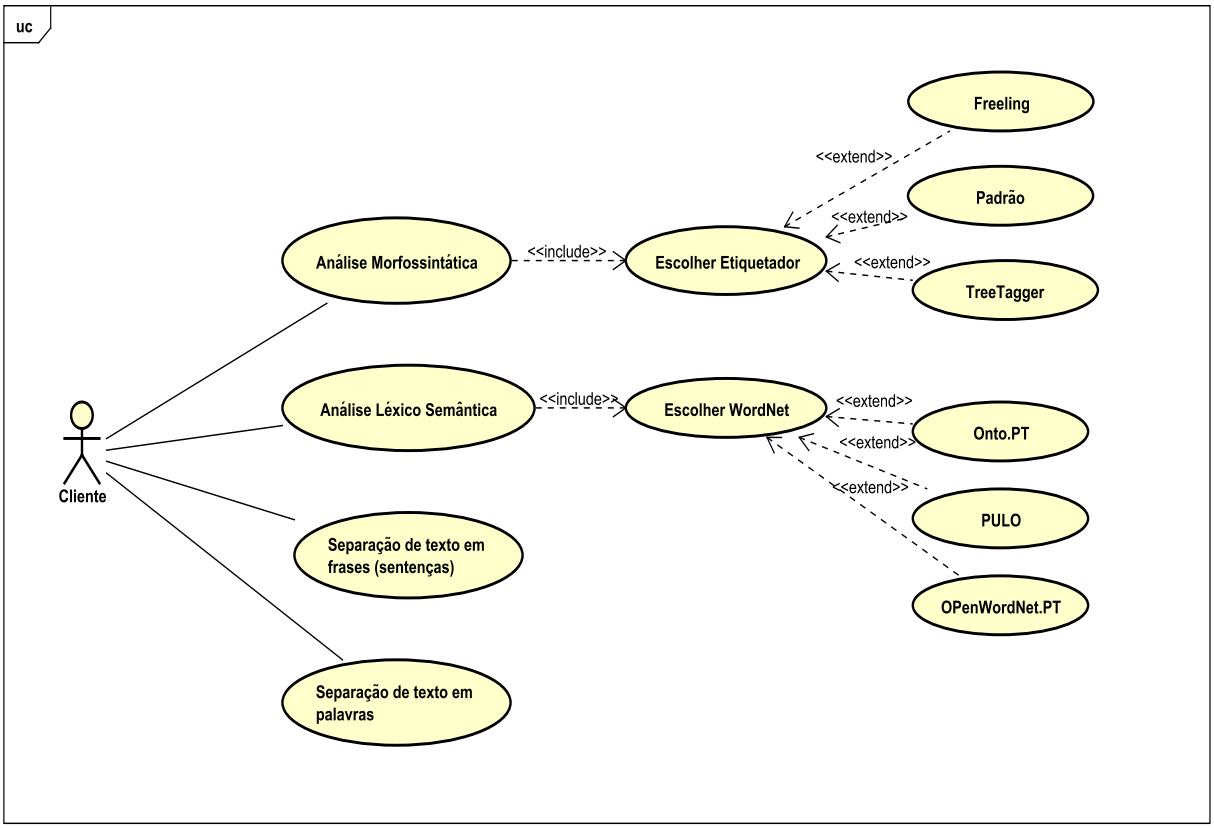

Figura 5.Diagrama de casos de uso para consultas via API.

Uma aplicação cliente pode requisitar qualquer um dos quatro serviços oferecidos pela plataforma, por meio do uso de chamadas HTTP GET. Por exemplo, um tradutor automático pode requisitar o serviço de análise léxico-semântica para

identificar os possíveis sentidos das palavras contidas no texto. Para isso, ele deve escolher uma dentre as três WNs disponíveis. Cada WN possui características diferentes. Assim, cabe a cada desenvolvedor das aplicações clientes conhecê-las e usá-las da melhor forma.

Ainda usando como exemplo um tradutor automático, ele pode usar os serviços de etiquetagem morfossintática para etiquetar palavras contidas em um texto, para aplicar alguma regra de tradução, segundo a distribuição sintática dessas palavras. Para isso, ele deve escolher um entre os três etiquetadores disponíveis. Cada etiquetador possui características diferentes, cabendo a cada desenvolvedor das aplicações clientes conhecê-las, para usá-las da melhor forma. Da mesma forma, usando os serviços da plataforma, um tradutor automático, para facilitar a tradução, pode dividir um texto de entrada em uma lista de sentenças ou mesmo em uma lista de palavras.

4.1.2.2 Cenário de uso no site do projeto

Basicamente o uso dos serviços, oferecidos no site do projeto PortService-Br, é feito por pesquisadores, estudantes, entre outros usuários, para conhecerem e testarem as ferramentas disponíveis na plataforma. A Figura 6 apresenta os casos de uso para esse cenário de consulta à plataforma.

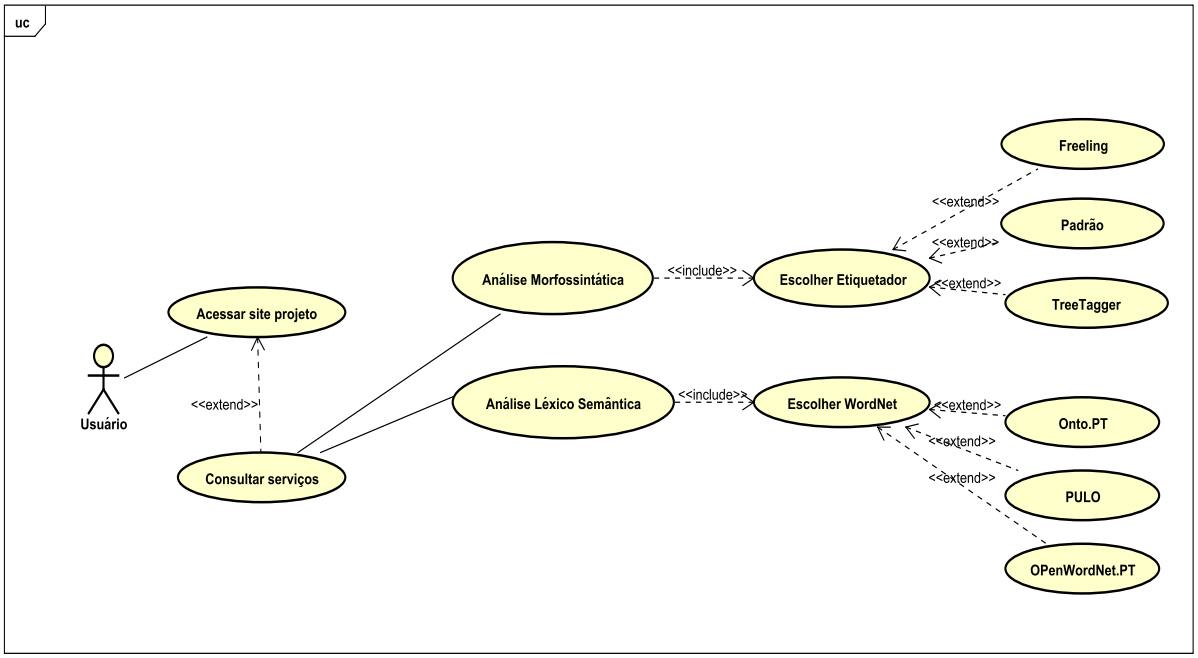

Figura 6. Diagrama de casos de uso para consultas no site do projeto.

Um usuário, seja um pesquisador, estudante ou qualquer outro tipo de usuário, tem pleno acesso às funcionalidades oferecidas na plataforma por intermédio de uma interface visual.

Um pesquisador em PLN, por exemplo, pode usar o serviço de análise léxico-semântica para identificar os possíveis sentidos de um conjunto de palavras. O pesquisador ainda pode escolher qual WN é mais indicada para responder às questões de sua pesquisa. O resultado da consulta é exibido na tela do site do projeto.

Um estudante pode aprender a classificar as palavras contidas em um texto (verbos, substantivos, adjetivos, advérbios, artigos, entre outros) usando os serviços de etiquetagem morfossintática para etiquetar palavras contidas nesse texto. O estudante seleciona um dos etiquetadores disponíveis ou usa o etiquetador padrão. O resultado é exibido na tela do site do projeto.

4.1.2.3 Cenários de uso de serviços do PortService-Br para a construção de um tradutor Português-Libras

Basicamente o uso dos serviços oferecidos no PortService-Br, na construção de um tradutor Português-Libras, é realizado mediante requisições HTTP para a API. O tradutor pode fazer uso de todos os serviços oferecidos na plataforma.

Para esse cenário foi desenvolvido uma API de tradução de Português-Libras, que usa a técnica de memória de tradução (ou tradução baseada em casos). O detalhamento da construção desse tradutor é apresentado no Capítulo 6. O tradutor usa cinco módulos para a tradução automática Português-Libras.

Os módulos implementam os seguintes procedimentos de tradução: 1) a mais simples procura por traduções anteriores comparando uma sentença inteira, 2) comparações de similaridade de expressões baseadas em synsets da WN e tags sintáticas, 3) tradução baseada em regras sintáticas, 4) tradução palavra por palavra e 5) tradução por datilologia.

A Figura 7 apresenta o caso de uso para o cenário de consulta à plataforma no primeiro módulo de tradução.

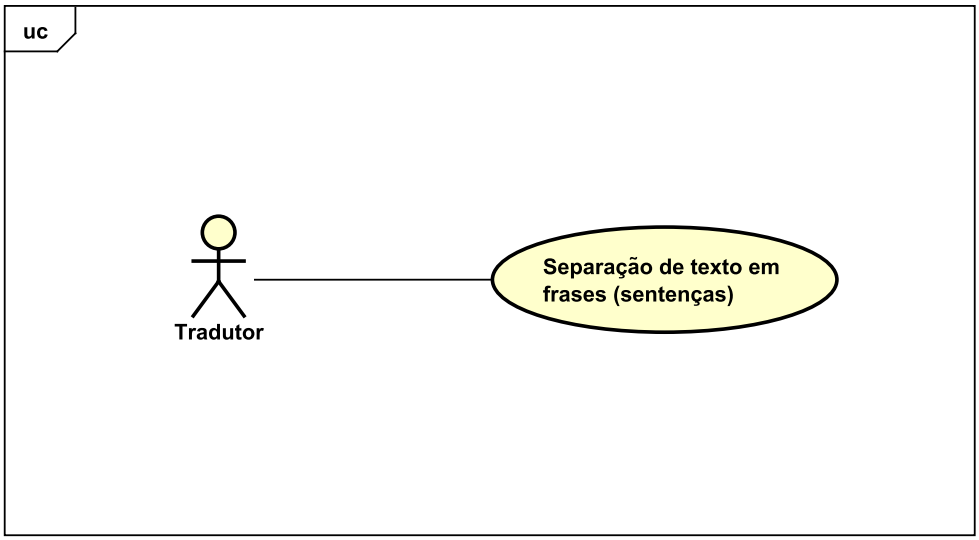

Figura 7. Caso de uso para o cenário de consulta à plataforma no primeiro módulo de tradução.

No primeiro módulo, a forma mais simples de tradução, um texto é convertido em uma sequência de sentenças ou frases. Neste módulo de tradução, cada sentença da lista de sentença, é dada como entrada para pesquisa direta. Consiste em pesquisar uma sentença exatamente igual à sentença de entrada na Memória de Tradução. A Figura 8 apresenta o caso de uso para o cenário de consulta à plataforma, de acordo com o segundo módulo de tradução.

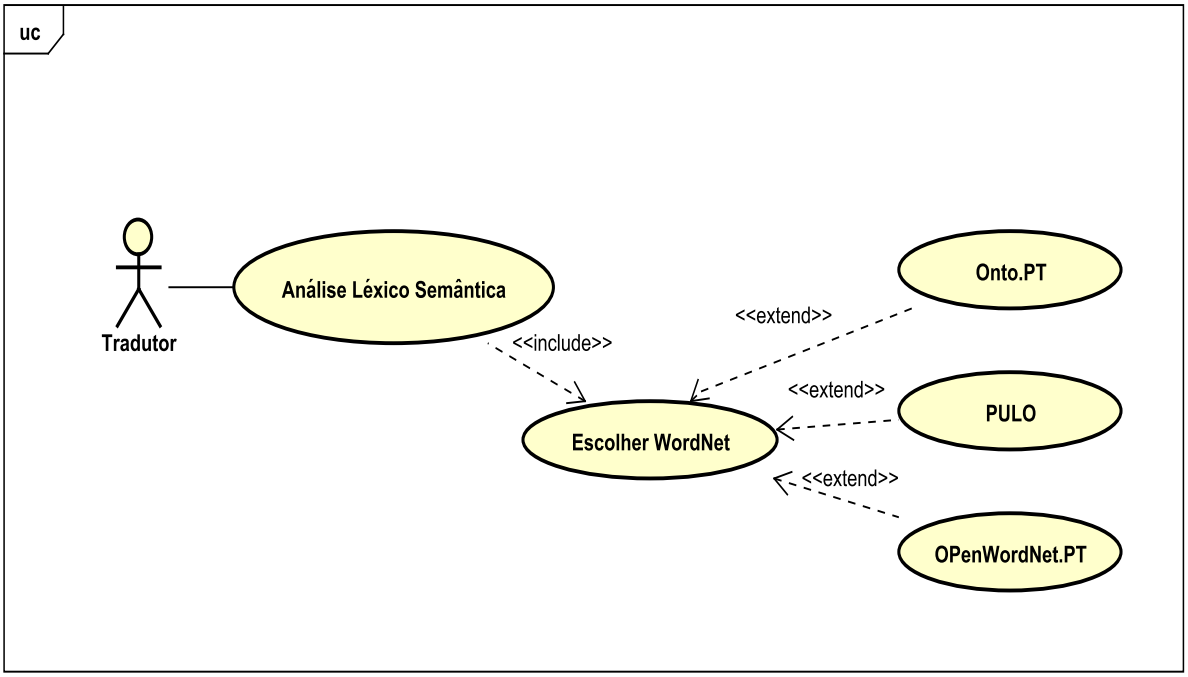

Figura 8. Caso de uso para o cenário de consulta à plataforma no segundo módulo de tradução.

No segundo módulo, primeiramente são obtidos os synsets e tags sintáticas de cada palavra do texto de entrada. Com as palavras etiquetadas com seus respectivos synsets e tags sintáticas é possível montar a expressão que a caracteriza. Essa expressão é dada como entrada para a pesquisa de expressões similares na memória de tradução. Essa estratégia parte do princípio de que, se existe uma expressão equivalente, existe uma tradução equivalente. A Figura 9 apresenta o caso de uso para o cenário de consulta à plataforma no terceiro módulo de tradução.

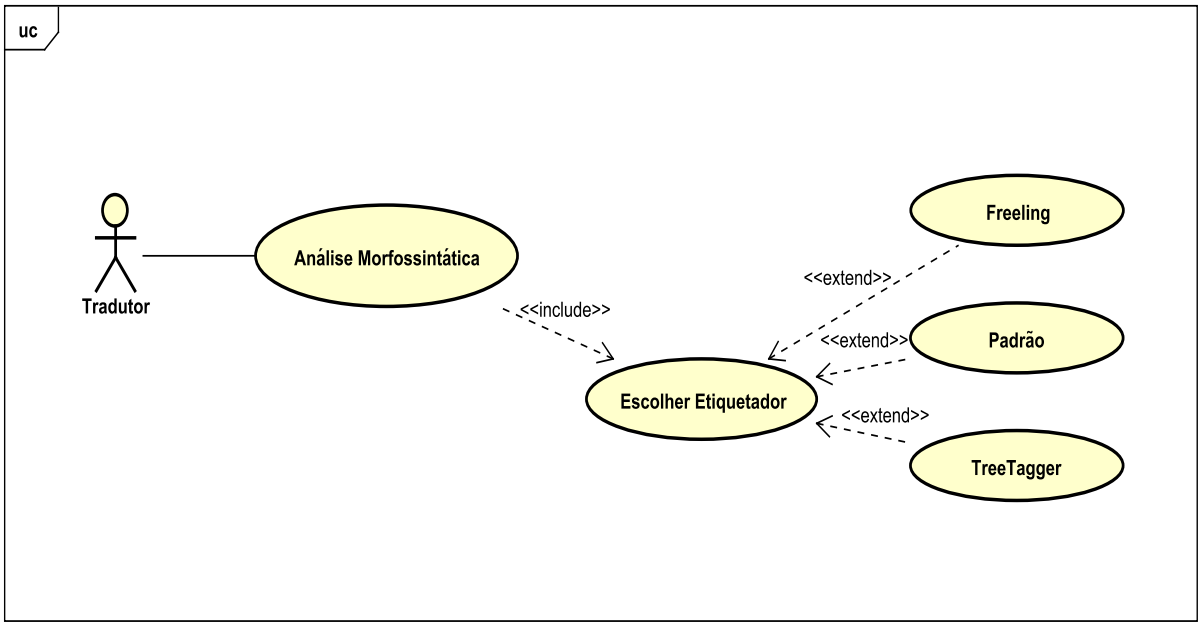

Figura 9. Caso de uso para o cenário de consulta à plataforma no terceiro módulo de tradução.

No terceiro módulo, primeiramente são obtidos uma lista de tags sintáticas das palavras do texto de entrada. Com a lista de tokens já etiquetados anteriormente, é possível montar a expressão sintática para a sentença de entrada. A expressão sintática é dada como entrada para uma pesquisa. Se existir uma regra que possua, na sua premissa, a expressão sintática de entrada, significa que é possível traduzir a sentença a partir de regras. A Figura 10 apresenta o caso de uso para o cenário de consulta à plataforma no quarto módulo de tradução.

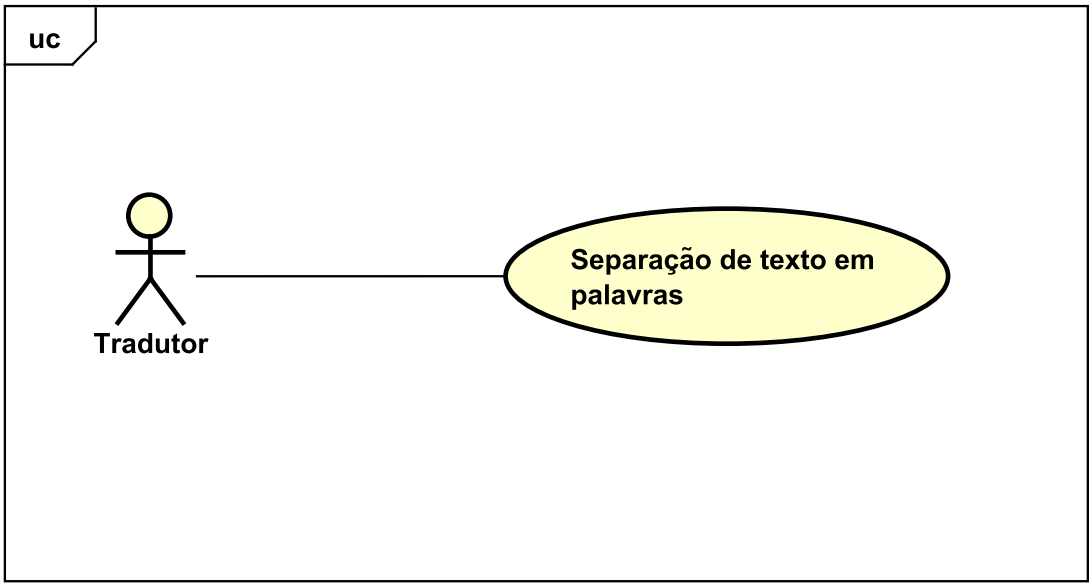

Figura 10. Caso de uso para o cenário de consulta à plataforma no quarto módulo de tradução.

O quarto módulo de tradução é acionado quando os três módulos anteriores não obtiveram êxito na tradução. É uma tradução mais simples feita palavra por palavra, sem levar em consideração nenhum tipo de contexto. O texto de entrada é dividido em uma lista de palavras que são dadas como entrada para uma pesquisa direta a um dicionário de tradução.

4.2 Projeto da Arquitetura

O PortService-Br é uma plataforma orientada a serviços de PLN para a língua portuguesa, que disponibiliza uma API para acesso a vários tipos de serviços. São disponibilizados até o momento ferramentas de etiquetagem morfossintática, recursos de análise léxico-semântica e segmentação de textos em sentenças (frases) e segmentação de textos em palavras (tokens).

Para a funcionalidade de etiquetagem morfossintática são usados a Freeling, ou a TreeTagger ou um etiquetador desenvolvido no âmbito deste projeto de dissertação. Essas ferramentas trabalham de modo independente, sendo necessário, durante a requisição do serviço, que seja especificado com qual delas a análise deve ser realizada. Para a análise léxico-semântica é usada a OpenWordNet-PT, a Onto.PT e a PULO.

A API REST foi desenvolvida com a linguagem de programação Python, escolhida devido aos recursos nativos de processamento de expressões regulares, programação dinâmica e por oferecer flexibilidade de implementação, tanto no paradigma procedural quanto no orientado a objetos. A Figura 11 apresenta a arquitetura do PortService-Br, de modo a lidar com os desafios apresentados na especificação de requisitos.

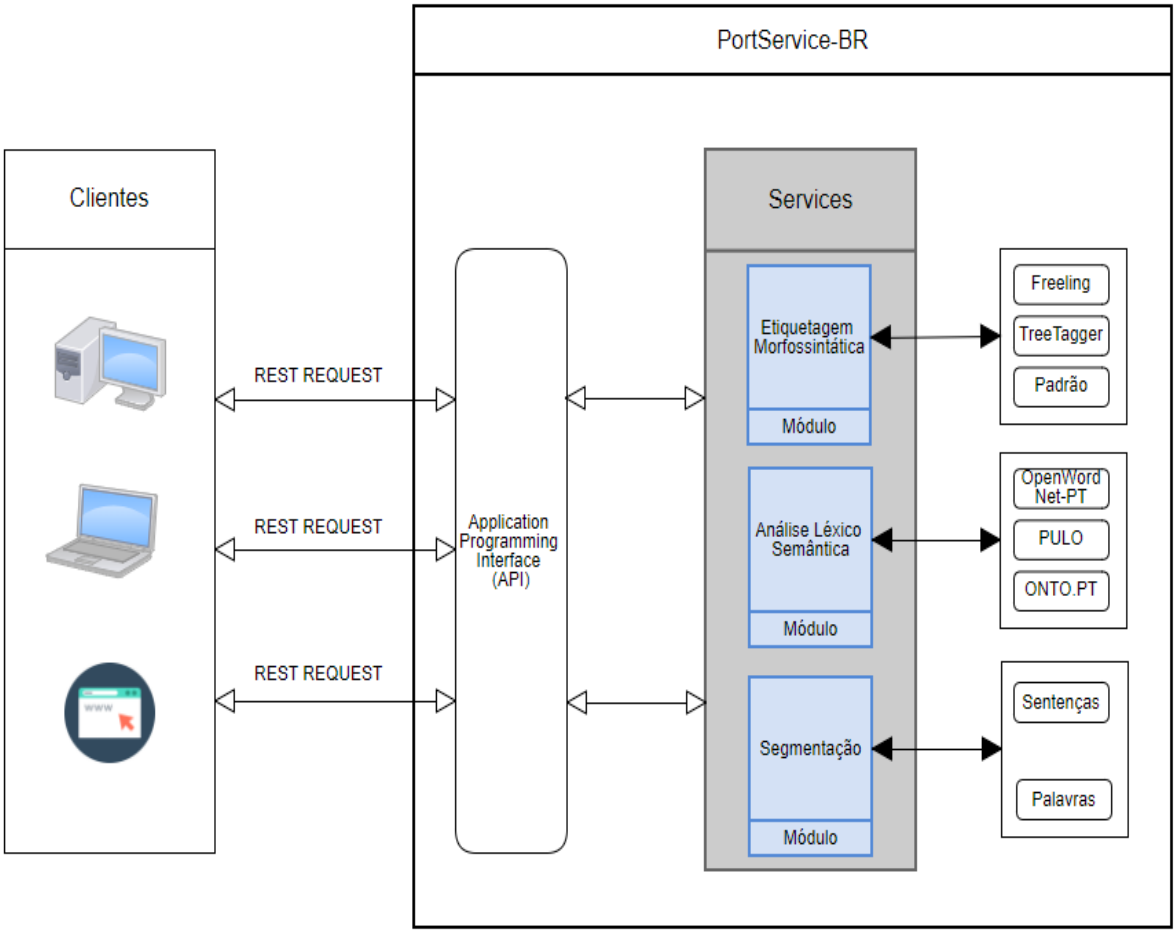

Figura 11. Arquitetura geral do PortService-Br.

O desafio inicial da arquitetura era a modelagem e criação de mecanismos flexíveis e extensíveis, para descrever as interfaces entre a API REST e as ferramentas disponibilizadas e instaladas no sistema, de modo a facilitar a adição de novas funcionalidades. A modelagem visou atender aos seguintes objetivos:

- oferta de um serviço modular, de modo a separar a modelagem de serviços diferentes;

- facilidade de adição de novas ferramentas ou funcionalidades;
- facilidade de comunicação entre cliente e servidor.

Clientes conectados à Internet realizam requisições aos serviços do PortService-Br através de simples URLs que indicam cada serviço. Basta apenas passar, como parâmetros, o texto ou palavra a ser analisada. O servidor então se encarrega de executar cada ferramenta requerida de modo independente.

4.2.1 Modelos de classes

A partir da análise do projeto da arquitetura e das funcionalidades requeridas, foi necessário especificar a modelagem conceitual de todo o sistema, ou seja, seu funcionamento interno após receber uma requisição remota. A Figura 12 apresenta o modelo de classes dos serviços de análise léxico-semântica e etiquetagem morfossintática da plataforma PortService-Br.

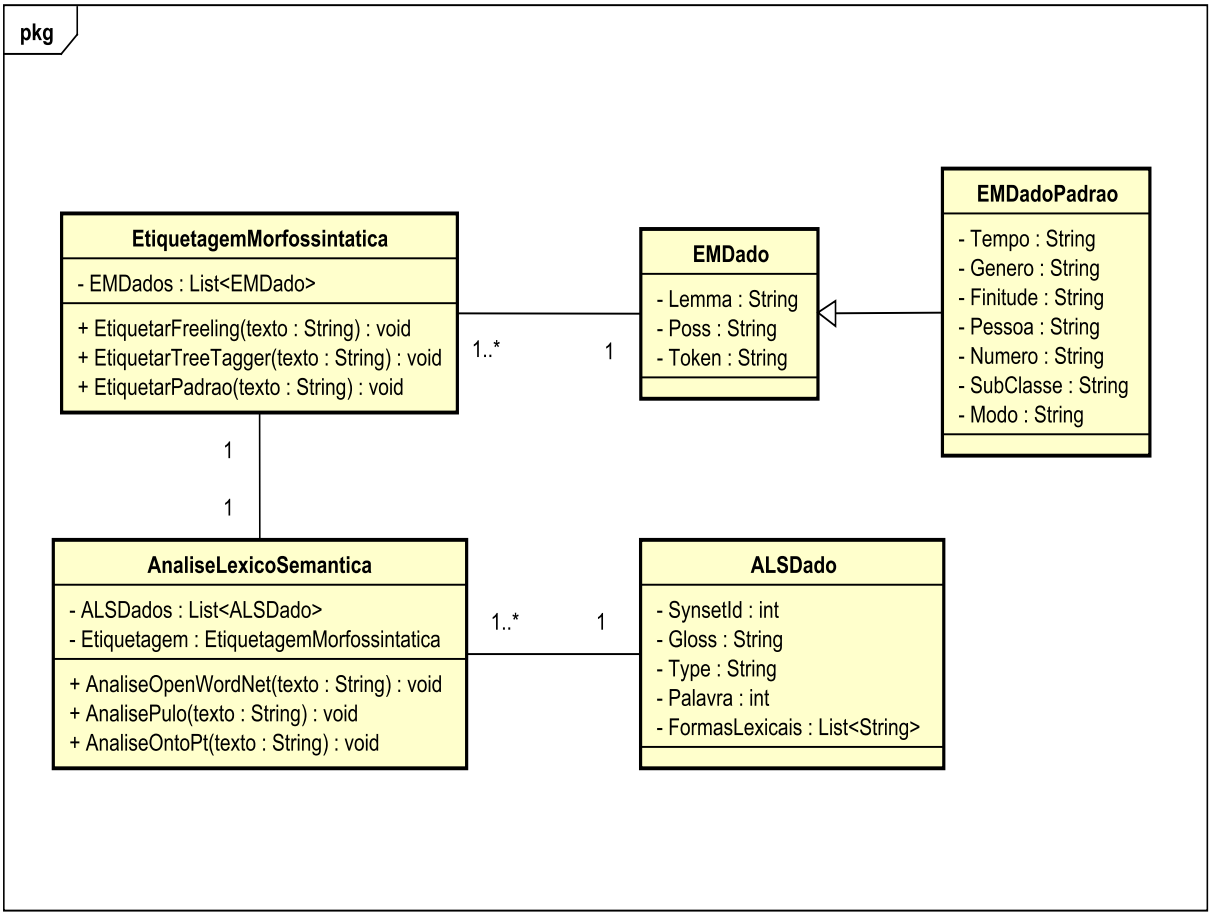

Figura 12. Modelo de classes dos serviços de análise léxico semântica e etiquetagem morfossintática do PortSevice-BR.

Classe EMDado: Descreve um objeto com um conjunto de propriedades de análise morfossintática de uma palavra. Lemma armazena a forma lematizada da palavra, ou seja, sem flexão em sua forma primitiva. Poss armazena classe gramatical, gênero e número de uma palavra. Token representa elementos de cada oração, que podem ser palavras, números e sinais de pontuação.

Classe EMDadoPradrao: Descreve um objeto, que herda da classe EMDado, é uma especialização para os requisitos de etiquetagem usando o etiquetador Padrão, contendo um conjunto específico de propriedades de análise morfossintática. Tempo é referente a verbos e representa o tempo verbal. Genero é referente ao gênero da palavra e se aplica a substantivos, adjetivos e determinantes. Pessoa é referente a verbos determina a pessoa da conjugação. Número é referente a substantivos, adjetivos e determinantes, podem estar no plural ou singular. Subclasse é referente a determinantes. Modo é referente a verbos, determina o modo verbal.

Classe ALSDado: Descreve um objeto com um conjunto de propriedades de uma análise léxico semântica, ou seja, armazena propriedades da WordNet de uma palavra. SynsetId armazena o identificador da wordnet. Gloss armazena uma descrição associada com o synset. Type armazena o tipo de synset, que pode ser dos tipos: verbo, advérbio, substantivo e adjetivo. Palavra armazena a palavra no seu estado normal. FormasLexicais representa todas as palavras associadas ao mesmo SynsetId, ou seja, com o mesmo significado.

Classe EtiquetagemMorfossintatica: É responsável por guardar a etiquetagem morfossintática de um texto ou palavra de entrada: A propriedade EMDados, armazena uma lista de EMDado gerados a partir do texto passado como parâmetro para um dos três métodos: EtiquetarFreeling que usa o etiquetador Freeling, EtiquetarTreeTagger que usa o etiquetador TreeTagger e EtiquetarPadrao que usa o etiquetador desenvolvido no âmbito do projeto PortService-Br.

Classe AnaliseLexicoSemantica: É responsável por gerar a análise léxico semântica de um texto ou palavra de entrada. A propriedade Etiquetagem, é responsável por receber o texto de entrada e gerar uma lista de objetos de EtiquetagemMorfossintatica. Dessa lista somente são usados os Lemmas de cada palavra, uma WordNet somente armazena os Lemmas das palavras, por isso a

necessidade do processo de lematização do texto de entrada. A propriedade ALSDados, armazena uma lista de itens de ALSDado, gerados à partir do texto de entrada passado como parâmetro a um dos três métodos: AnaliseOpenWordNet que usa a WordNet OpenWordNet-PT, AnalisePulo que usa a WordNet PULO e AnaliseOntoPt que usa a WordNet ONTO.PT. A Figura 13 apresenta o modelo de classes dos serviços de segmentação de texto em sentenças e segmentação de textos em palavras da plataforma PortService-Br.

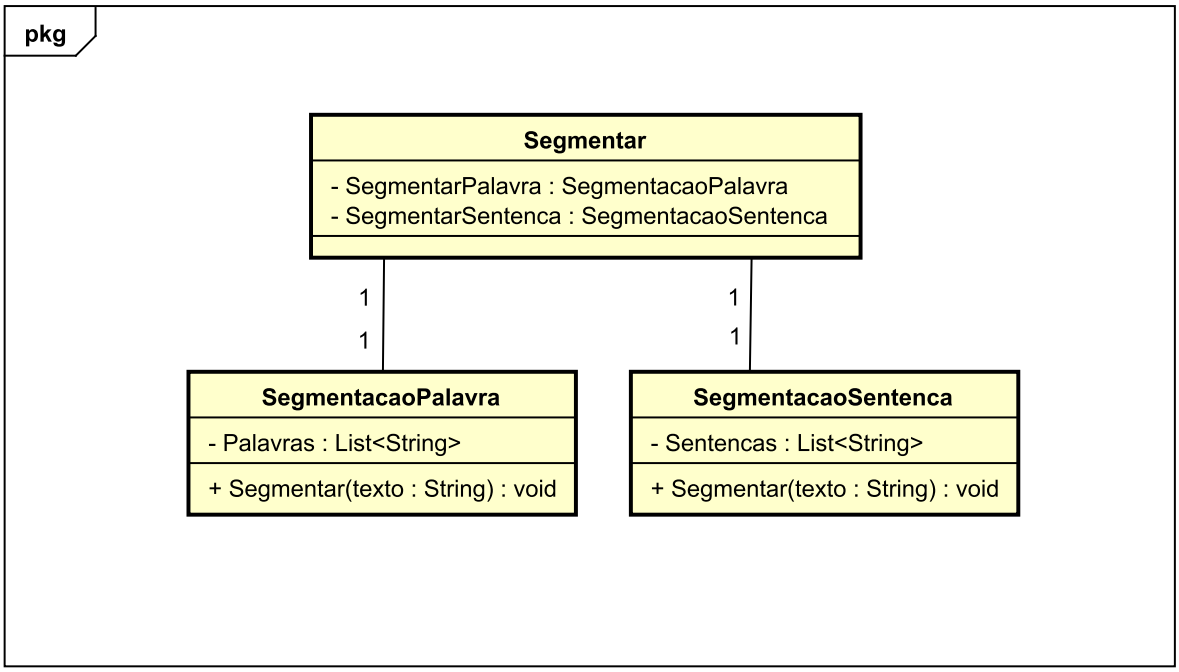

Figura 13. Modelo de classes segmentação de sentenças e segmentação de palavras.

Classe SegmentacaoSentenca: Essa classe é responsável por segmentar um texto em um conjunto de sentenças (frases). A propriedade sentença, armazena uma lista de sentenças geradas à partir do texto passado como parâmetro para o método Segmentar.

Classe SegmentacaoPalavra: Essa classe é responsável por segmentar um texto em um conjunto de palavras. A propriedade Palavras armazena uma lista de palavras (no caso também pontuações) a partir do texto passado como parâmetro para o método Segmentar.

Classe Segmentar: Essa classe é uma classe repositório para chamadas a classes

específicas para as chamadas às classes SegmentacaoSentenca e SegmentacaoPalavra.

4.3 Fluxo do servidor ao receber requisições

Ao receber uma requisição de serviço o servidor deve se encarregar de tratá-la corretamente. A Figura 14 apresenta o fluxo de tarefas do servidor ao receber uma chamada ao serviço de etiquetagem morfossintática.

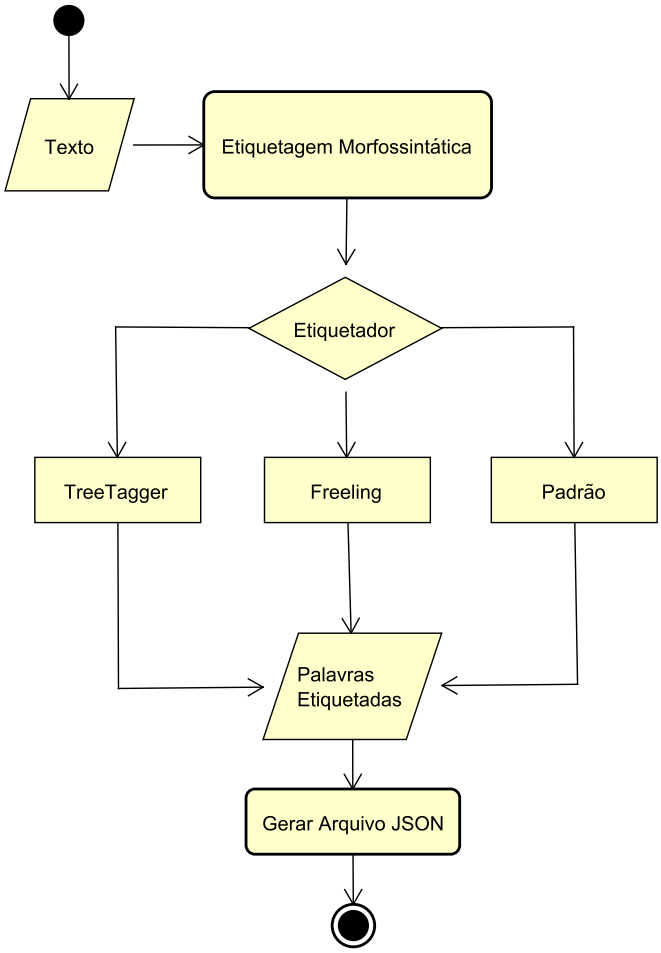

Figura 14. Fluxo de tarefas do servidor ao receber uma chamada ao serviço de etiquetagem morfossintática.

Um texto é dado como entrada para o módulo de etiquetagem morfossintática, juntamente com o parâmetro que especifica qual etiquetador deve ser usado para gerar a etiquetagem. Ao final do processo de etiquetagem a saída é uma lista de palavras etiquetadas que é passada como entrada para o módulo que gera o arquivo *json* de saída. O arquivo *json* é então retornado para o requisitante do serviço. A

Figura 15 apresenta o fluxo de tarefas do servidor ao receber uma chamada ao serviço de análise léxico semântica.

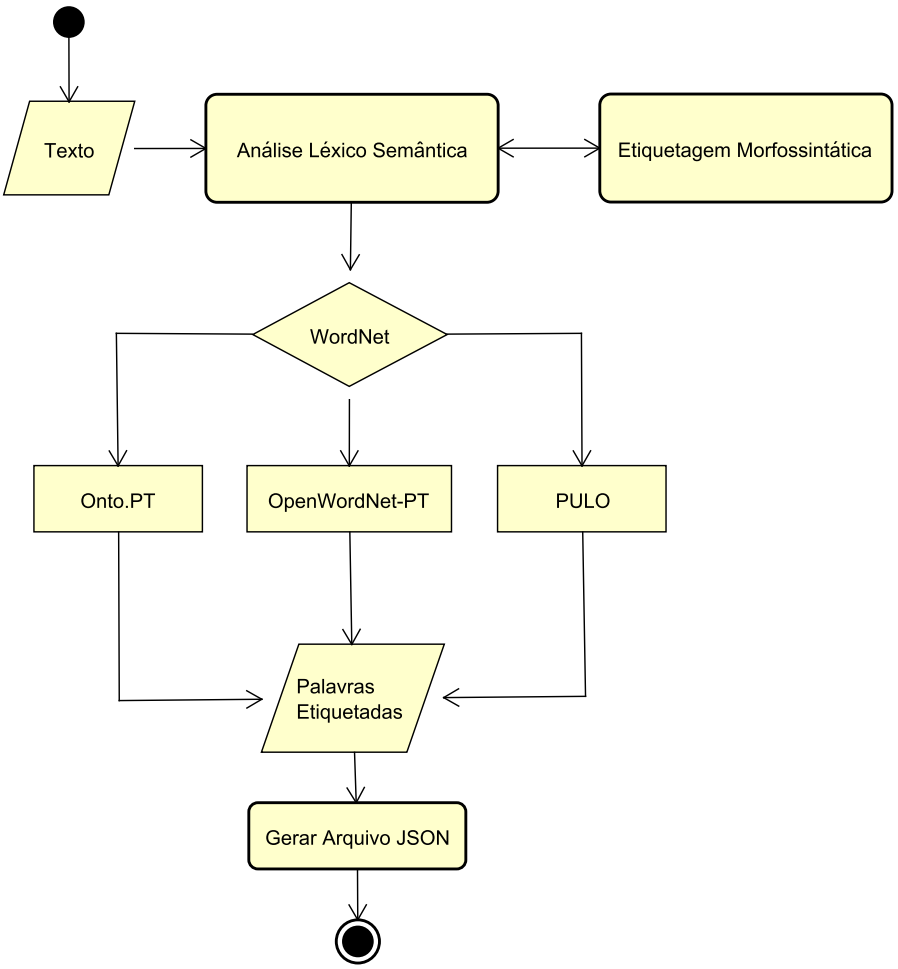

Figura 15. Fluxo de tarefas do servidor ao receber uma chamada ao serviço de análise léxico-semântica.

Um texto é dado como entrada para o módulo de análise léxico semântica, juntamente com o parâmetro que indica qual a *Wordnet* será usada. Primeiramente o texto é repassado para o módulo de etiquetagem morfossintática para obter o lemma referente a cada palavra do texto, sendo então retornada uma lista de palavras etiquetadas. Em uma WordNet são armazenados somente os lemas das palavras, ou seja, não existem flexões, por este motivo é imprescindível que as palavras contidas no texto de entrada sejam lematizadas. Essa lista é então repassada para a WordNet escolhida. Ao final temos uma lista de palavras etiquetadas com os sentidos da *WordNet*, essa lista é então passada como entrada para o módulo que gera o arquivo *json* de saída. O arquivo *json* é então retornado

para o requisitante do serviço.

A Figura 16 apresenta o fluxo de tarefas do servidor ao receber uma chamada ao serviço de segmentação.

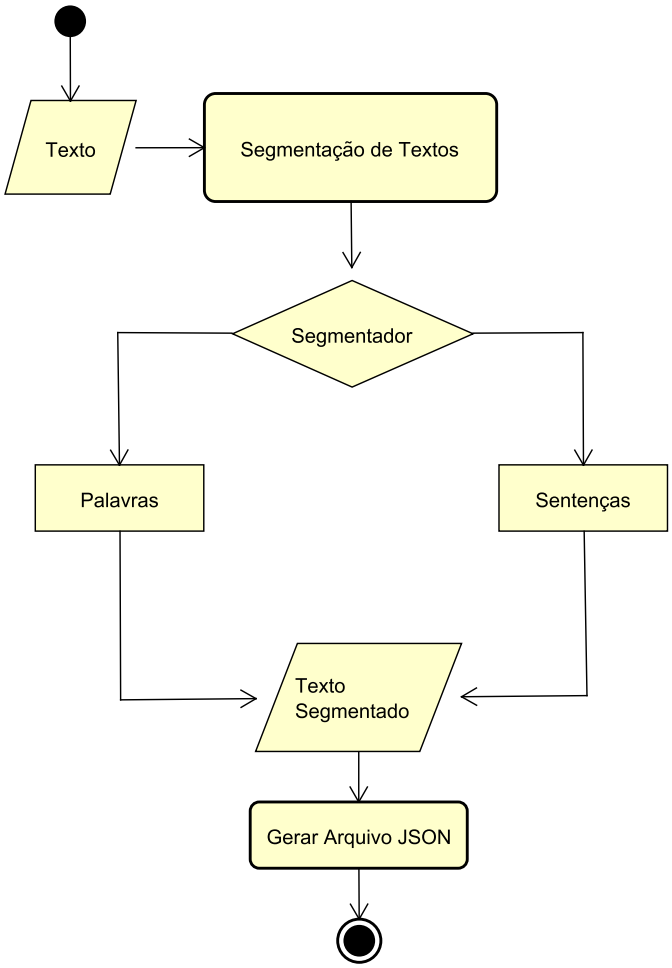

Figura 16. Fluxo de tarefas do servidor ao receber uma chamada ao serviço de segmentação de texto.

Um texto é dado como entrada para o módulo de segmentação juntamente com o parâmetro que indica qual segmentação deve ser realizada. Ao final temos uma lista de palavras ou uma lista de sentenças, essa lista é então passada como entrada para o módulo que gera o arquivo json de saída. O arquivo json é então retornado para o requisitante do serviço.

5. IMPLEMENTAÇÃO DO PORTSERVICE-BR

Neste capítulo são apresentados os detalhes da implementação da plataforma PortService-Br, suas interfaces de consulta e detalhes de seu funcionamento. Após a definição da arquitetura foi necessário proceder com a implementação de um protótipo e sua disponibilização na rede mundial de computadores.

5.1 Tecnologias utilizadas

De modo a tornar a implementação mais rápida, confiável e eficiente, foi preciso pesquisar linguagens de programação e respectivos *Frameworks* que facilitassem o processo de desenvolvimento do protótipo. As pesquisas culminaram na escolha da linguagem de programação *Python*, e os respectivos *Frameworks* Django e Django *Rest Framework*, e da biblioteca de processamento de linguagem natural NLTK.

5.1.1 Python

Python é uma linguagem de programação com código aberto, foi desenvolvida na Holanda, por Guido van Rossum, nos anos 1980. Até hoje o pesquisador é um dos principais desenvolvedores e concentrando todas as decisões importantes dos rumos do desenvolvimento. *Python* frequentemente é vista como uma linguagem do tipo script, porém se trata de uma linguagem de propósito geral, usada tanto para elaborar simples scripts quanto para grandes servidores Web escaláveis.

Seu uso também se estende para programação com interfaces de usuários gráficas, bancos de dados, sites Web, tanto do lado cliente quanto do lado servidor, e também para testes de aplicações. Essa linguagem pode ser usada para o desenvolvimento de programas de sistemas de simulação que executam nos supercomputadores mais rápidos do mundo ou para crianças aprenderem as suas primeiras lições (VAN ROSSUM, 2015).

Uma de suas principais características são a tipagem dinâmica (*dynamic typing*). Em Python, variáveis são apenas nomeadas e se referem a objetos. Não é necessária sua declaração antes de serem utilizadas e podem mudar seu tipo ao longo do programa (VAN ROSSUM, 2015). Mas o principal diferencial da linguagem é o uso do recurso para agrupar blocos de código, sem a necessidade do ponto e vírgula para separação de linhas.

Python tem grande facilidade na criação de pacotes. Um pacote é uma forma de agrupar código de forma reutilizável. Este fato facilita, juntamente com um gerenciador de pacotes, o compartilhamento de bibliotecas e soluções com a grande comunidade da linguagem. Python está entre as cinco linguagens mais populares no ano de 2015 (CASS, 2015).

5.1.2 Django

Trata-se de um *Framework* com código aberto para desenvolvimento Web que usa *Python* como linguagem de programação. Foi criado em 2003 por desenvolvedores Web de um jornal americano, o *framework* atualmente é mantido e atualizado pela Django Software Foundation, que é de uma organização americana independente e sem fins lucrativos. Dentre os sites conhecidos que fazem uso de Django no seu desenvolvimento podemos citar: NASA, Instagram, Pinterest e Mozilla (DJANGO SOFTWARE FOUNDATION, 2015).

Django tem suporte nativo a uma grande variedade de necessidades comuns em se tratando de aplicações Web, tanto no nível de desenvolvimento (upload de arquivos, geração de CSV (*Comma-Separated Values*), mapeamento automático entre classes do modelo de dados e formulários, autenticação, cache, logs, sessão), quanto em administração, disponibilizando um site completo de gerenciamento de usuários, senhas, acessos, entre outras funcionalidades (DJANGO SOFTWARE FOUNDATION, 2015). O *Framework* segue fortemente o padrão MVC (*Model-View-Controller*), dividindo os componentes da aplicação em três papéis distintos: *model*, responsável pelos dados e pela lógica de negócio; *view*, que é a camada de apresentação dos dados; e *controller*, que manipula a *model* e controla a atualização da *view*.

Na camada de modelo de dados (*model*), classes em *Python* definem um modelo de dados e o ORM próprio do framework faz o mapeamento para um banco de dados relacionais, que pode ser PostgreSQL, MySQL, SQLite ou Oracle. Todos esses bancos de dados são nativamente suportados, entre outros reconhecidamente compatíveis.

A camada de apresentação (*view*) é implementada em um sistema baseado em templates; código estático escrito em linguagem de marcação HTML pode ser

escrito, em conjunto com diretivas, possibilitando assim inclusão de conteúdo dinâmico, além de permitir que se controle o fluxo de execução a partir desses trechos (DJANGO SOFTWARE FOUNDATION, 2015).

A camada de controle (*controller*) conhecida como view. Na interpretação dos desenvolvedores do framework, “a view pode descrever um conjunto de informações apresentado para o usuário. Não necessariamente como uma informação, mas qual informação é apresentada. A view descreve qual informação você vê, e não como você vê” (DJANGO SOFTWARE FOUNDATION, 2015). Assim, views em Python são funções que podem responder a determinada URL. Essas URLs são roteadas por módulos baseados em expressões regulares, que interpretam as requisições e as direcionam para as *views* responsáveis por respondê-las (DJANGO SOFTWARE FOUNDATION, 2015).

5.1.3 Django Rest Framework

Django REST *Framework* é um dos mais conhecidos *frameworks* para implementação de APIs do tipo REST em conjunto com o Django. Prove um grande número de funcionalidades (roteamento, serialização, validação, autenticação, permissões, filtros, paginação, versionamento, entre outros). Também permite uma vasta gama de variação das configurações dessas funcionalidades de forma intuitiva, este fato permite implementar completas soluções em poucas linhas de código. Os principais componentes do framework são os seguintes:

- ViewSets: similar ao que outros frameworks chamam de Resource ou Controllers, permite que se combine um conjunto de views relacionadas em uma única classe.

- Métodos como get, post, list ou retrieve estarão todos disponíveis na mesma classe, e as URLs de roteamento serão geradas automaticamente (CHRISTIE, 2015).

- Routers: responsável por gerar as URLs para cada recurso. Pode fazer isso automaticamente para ViewSets (CHRISTIE, 2015).

- Parsers: trata a requisição recebida baseado no tipo de mídia informado. Possui parsers para JSON e upload de arquivos, entre outros (CHRISTIE,

2015).

- Renderers: define o formato em que a requisição será respondida. Possui renderers para JSON, HTML e até mesmo para uma página de demonstração detalhada da API e de cada recurso (CHRISTIE, 2015).

- Serializers: responsáveis pela serialização e desserialização, permitindo a transformação entre objetos complexos como instâncias de classes da model em classes Python com tipos nativos, que podem ser renderizadas facilmente em JSON ou XML por exemplo (CHRISTIE, 2015).

- FilterSets: o comportamento padrão das views do tipo list é retornar todas as instâncias de uma determinada model. Filtros permitem limitar os resultados retornados, e os FilterSets permitem definir um conjunto de filtros que uma view deve usar (CHRISTIE, 2015).

5.1.3 Natural Language Toolkit (NLTK)

Criado originalmente em 2001 como parte de um curso de linguística computacional do Departamento de Ciência da Computação e Informação, da Universidade da Pensilvânia, o NLTK é uma plataforma usada para construir programas Python que trabalhem com dados de linguagem humana para aplicação em PLN (Perkins, 2014).

O NLTK define uma infraestrutura que pode ser usada para construir programas de PLN em Python; fornece classes básicas para representar dados relevantes para o processamento da linguagem natural; interfaces padrão para executar tarefas como tokenização,etiquetagem morfossintática, análise sintática e classificação de texto; e implementações padrões para cada tarefa que podem ser combinadas para resolver problemas complexos (Perkins, 2014).

NLTK é um conjunto de bibliotecas para PLN que se tornou uma das melhores ferramentas para prototipagem e construção de sistemas de PLN. A opção pelo NLTK teve por base os critérios de facilidade de uso, produtividade e amplitude de cobertura não só da biblioteca em si, mas também da linguagem de programação Python. A biblioteca fornece um *framework* unificado, com interfaces e estruturas de dados consistentes e nomes de métodos de fácil compreensão (Perkins, 2014).

5.2 Desenvolvimento do etiquetador padrão

Apesar dos etiquetadores *Freeling* e *TreeTagger* serem boas ferramentas de etiquetagem, fornecem um conjunto restrito de informações em cada tag. Para contar com um número maior de informações disponíveis de forma individualizada foi desenvolvido um etiquetador morfossintático com o auxílio da biblioteca NLTK, que contém um conjunto maior de informações disponíveis de forma individual.

Para o desenvolvimento do etiquetador foi usado o (NLTK). Para o treinamento do etiquetador foram usadas os primeiras 16 milhões de tokens do *corpus* CETEMFolha (*Corpus* de Extratos de Textos Eletrônicos/Folha de São Paulo), por motivos de restrições de *hardware* não foi possível usar todo o *corpus*, não havia memória RAM suficiente para comportar todo o *corpus*. O *corpus* CETEMFolha possui cerca de 24 milhões de palavras em português brasileiro. Esse *corpus* foi criado no projeto Linguateca, no Núcleo Interinstitucional de Linguística Computacional de São Carlos. O *corpus* é etiquetado pelo *parser* Palavras, e está disponível no site da Linguateca[4] em formato de arquivo texto, entre outros formatos.

Foi desenvolvido um algoritmo capaz de percorrer esses arquivos em formato de texto e extrair as propriedades necessárias para o treinamento do etiquetador (flexão, lemma e propriedades morfossintáticas). As propriedades morfossintáticas são extraídas das *tags* de classes de palavras, de inflexão e de subclasse. Todas as propriedades foram armazenadas em uma base de dados.

Durante a execução do algoritmo são identificadas expressões multipalavras (palavras compostas), que além de serem armazenadas na base de dados, também são inseridas em um dicionário. As palavras compostas ou expressões multipalavras são combinações de dois ou mais lexemas. A identificação dessas expressões representa um gargalo para aplicações de PLN, principalmente para etiquetadores morfossintáticos. A construção de recursos capazes de realizar essa identificação pode se tornar onerosa. Assim, uma alternativa atraente tem sido a extração automática delas (Rondon, Caseli e Ramisch, 2015).

[4] http://www.linguateca.pt/

Ao final da extração das propriedades dos *tokens* de cada linha, o algoritmo retorna uma tupla contendo a palavra e um *id* de suas propriedades morfossintáticas. Essa tupla é então armazenada em um vetor, que é usado para o treinamento do etiquetador. Cada registro da base de dados, compõe o *tagset* (conjunto de tags). A Tabela 6 apresenta o quantitativo de lemmas, flexões e propriedades morfossintáticas, após o término do algoritmo.

Tabela 6. Quantitativo de lemas, flexões e propriedades morfossintáticas.

Lemas	Flexões	Propriedades
285073	356331	272

Etiquetadores convencionais utilizam o esquema de tupla com (*token*, *tag*) tanto para o treinamento quanto para a etiquetagem (Perkins, 2014). Essa *tag* em geral representa apenas a classe gramatical (Pos) da palavra, não sendo possível acessar outras informações importantes como gênero, número, tempo verbal, entre outras.

5.2.1 Treinamento do Algoritmo de Etiquetagem

Com o vetor de treinamento, gerado após a extração das propriedades do *corpus*, é possível realizar o treinamento do etiquetador. É usado um *tagset* com 272 tags, correspondentes ao total de registros gerados na base de dados. A tarefa de POS *tagging* pode ser formulada como segue: dada uma palavra *Wj* derivada de uma sequência de palavras (*Wi*…*Wn*}, atribuir a melhor *tag Ti*, derivada do conjunto de *tags T= {Ti...Tn}*.

Para o treinamento foram selecionados três algoritmos da biblioteca NLTK, que possibilitam a construção de etiquetadores morfossintáticos, baseados, respectivamente, em unigramas (*UnigramTagger*), bigramas (*BigramTagger*) e trigramas (*TrigramTagger*). Todos os algoritmos são de aprendizagem estatística e supervisionada.

Os algoritmos são treinados usando a abordagem de encadeamento sequencial de etiquetadores, chamado de *sequential backoff tagger*. Primeiramente, é treinado um etiquetador *UnigramTagger*. Em seguida é treinado um etiquetador *BigramTagger*,

que recebe como *backoff* o etiquetador *UnigramTagger*. Por fim é treinado um algoritmo *TrigramTagger*, que recebe como *backoff* o etiquetador *BigramTagger* (Perkins, 2014).

O *sequential backoff tagger* cria uma cadeia de algoritmos POS *tagging*, que são executados de forma sequencial quando recebem um token de entrada. O encadeamento funciona de tal forma que, se o *token* analisado não for conhecido pelo algoritmo *TrigramTagger* é acionado o algoritmo *BigramTagger*. Essa lógica de encadeamento se repete para cada par de etiquetadores, garantido que se um etiquetador é incapaz de classificar um *token*, esse mesmo *token* é passado ao algoritmo seguinte (Perkins, 2014).

5.2.2 Aplicação da Transformada de Brill

A transformada de Brill é um exemplo de Transformação Baseada em Aprendizagem Dirigida por Erro (TBL), desenvolvida por (Brill, 1992). Assim como simples etiquetadores estatísticos, ela começa por etiquetar palavras com suas marcas mais prováveis. A etiquetagem pode ser realizada por simples algoritmos baseados em n-gramas ou ainda uma combinação deles. Em seguida, o algoritmo observa quais etiquetas foram marcadas incorretamente e tenta induzir regras automaticamente, através de modelos sensíveis ao contexto. Finalmente, ele remarca o corpus de acordo com os padrões aprendidos (Brill, 1992), (Domingues, 2011) e (Danso e Lamb, 2014).

A transformada de Brill requer como entrada algoritmos estatísticos ou uma combinação deles. Como entrada para o algoritmo de transformação de Brill, foi atribuído o algoritmo que utiliza técnicas de *backoff* mencionado na seção anterior. Para a aplicação da transformada foi usado o algoritmo *BrillTagger* da biblioteca NLTK.

Foram utilizados os parâmetros padrão de todos os algoritmos. Foi definido um número mínimo de 300 regras a serem aprendidas automaticamente a partir do *corpus* de treinamento. Ao final desse experimento, tem-se como saída um algoritmo híbrido que contempla métodos estatísticos e regras induzidas de forma automática.

O Quadro 1 exibe as 3 primeiras regras inferidas pelo algoritmo de Brill. As regras produzidas exemplificam a capacidade do algoritmo de Brill em inferir regras sintáticas, a partir de erros de etiquetadores estatísticos, baseando-se nas posições das *tags*.

Quadro 1. Primeiras regras inferidas pelo algoritmo de Brill.

1. 17 → 2 if the Pos of the following word is "27".
2. 23 → 38 if the Pos of the following word is "33".
3. 35 → 18 if the Pos of the following word is "20".

A avaliação definida em cada regra verifica a posição atual do *corpus* para um determinado conjunto de condições, se todo o conjunto de condições de uma regra for atendido, a regra é acionada, isso implica na mudança de uma *tag* A para uma *tag* B. A Figura 17 apresenta toda a arquitetura de treinamento do etiquetador híbrido aqui proposto.

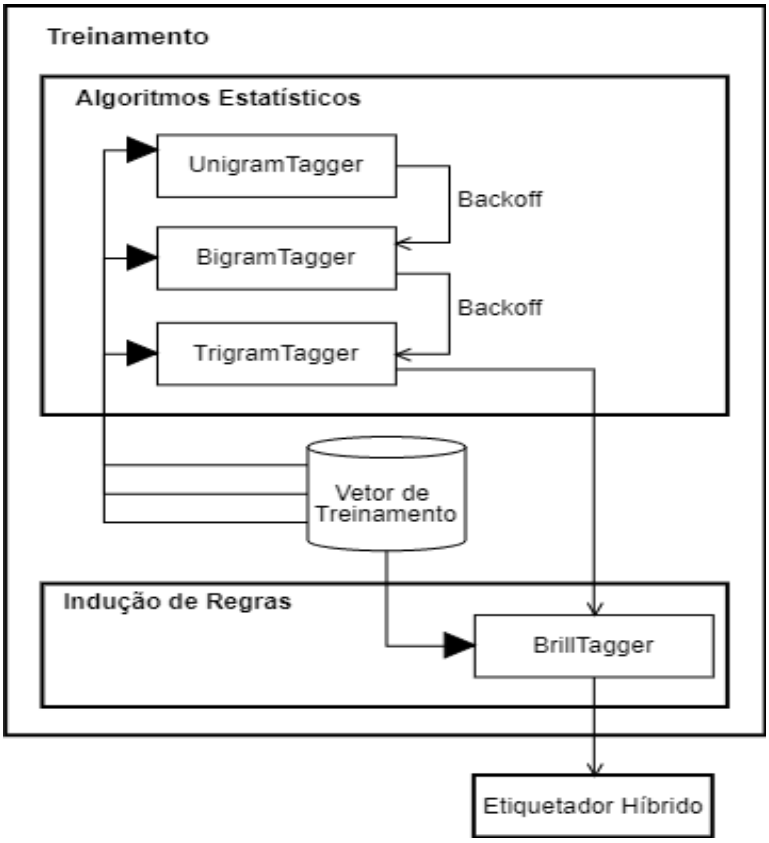

Figura 17. Arquitetura de treinamento do etiquetador híbrido.

Dessa forma, caso o algoritmo *BrillTagger* não consiga etiquetar uma palavra usando sua base de regras, será acionado o encadeamento sequencial de etiquetadores *backoff* de algoritmos estatísticos apresentados na seção anterior. Essa técnica diminui muito as chances de uma palavra não ser etiquetada. Algoritmos com arquitetura estatística somente conseguem etiquetar palavras que tenham sido usadas durante o treinamento. Com o auxílio de regras, também é possível etiquetar palavras desconhecidas.

5.2.3 Avaliação e testes dos algoritmos

Para fins de validação desse experimento, foram realizados testes de acurácia na tarefa de etiquetagem dos algoritmos *UnigramTagger*, *BigramTagger*, *TrigramTagger*, do algoritmo *sequential backoff tagger* e do algoritmo híbrido gerado com a transformada de Brill sobre o algoritmo *sequential backoff tagger.* Os testes foram realizados com algoritmos da própria NLTK.

Todos os algoritmos foram treinados usando o mesmo conjunto de dados e o mesmo *tagset*. Foram selecionados 80% dos dados contidos no vetor, cerca 12.800 milhões de *tokens*, para treinamento dos algoritmos, e 20%, cerca de 3.200 milhões de *tokens* para a realização dos testes. A Tabela 7 apresenta os resultados da acurácia de cada um dos algoritmos *UnigramTagger*, *BigramTagger* e *TrigramTagger*.

Tabela 7. Acurácia dos algoritmos, BigramTagger e TrigramTagger.

Algoritmo	Acurácia (%)
UnigramTagger	90,06
BigramTagger	43,07
TrigramTagger	22,20

Como pode ser visualizado na Tabela 7, o algoritmo *UnigramTagger* com 90,06% de acurácia teve melhor desempenho em relação ao algoritmo *BigramTagger*, com 43,07%, e ao *TrigramTagger*, com 22,20%, quando aplicados isoladamente.

Em comparação aos algoritmos analisados de forma isolada, logicamente a probabilidade de se encontrar trigramas e bigramas em conjunto em diferentes pontos do texto é inferior à de encontrar unigramas. Os resultados da Tabela 7, mostram que, de forma isolada, algoritmos n-gramas têm um baixo índice de acurácia.

A combinação desses algoritmos, com técnicas de encadeamento de etiquetadores, torna sua precisão maior do que a de qualquer etiquetador individual. A Tabela 8 apresenta os resultados da acurácia das fases de encadeamento de etiquetadores. A Tabela 8 também apresenta a acurácia do etiquetador híbrido que aplica o algoritmo de Brill sobre o algoritmo final gerado pelo encadeamento de etiquetadores.

Tabela 8. Acurácia do encadeamento de etiquetadores e do algoritmo híbrido.

Algoritmo	Acurácia (%)
UnigramTagger	90,06
BigramTagger + (uni)	91,57
TrigramTagger + (big)	93,29
Brill	97,29

Como pode ser visualizado na Tabela 8, a acurácia do algoritmo *UnigramTagger* continua com o mesmo percentual 90,06%. A precisão do algoritmo *BigramTagger*, tendo como *backoff* o algoritmo *UnigramTagger*, aumenta em mais de 1% a acurácia, elevando o seu percentual de acurácia para 91,57%. Por fim, esse percentual é elevado em mais de 2% quando o algoritmo *BigramTagger* é passado como *backoff* para o algoritmo *TrigramTagger*, elevando sua acurácia para 93,29%.

O algoritmo híbrido que aplica o algoritmo de Brill sobre o algoritmo final gerado pelo encadeamento, obteve acurácia de 97,90%. A Figura 18 apresenta de forma gráfica o percentual de aumento da acurácia em cada fase de encadeamento.

Os resultados apresentados mostram que a técnica de encadeamento de etiquetadores, em conjunto com técnicas de TBL como a de Brill, podem elevar bastante o percentual de acurácia de etiquetadores. Esses resultados ressaltam a eficácia da técnica proposta. A precisão desse tipo de etiquetador pode cair ou mesmo aumentar acentuadamente dependendo do tema, época ou estilo de escrita e da formação do corpus exposto para treinamento (Manning, 2011). Os bons resultados obtidos com o algoritmo híbrido, gerado a partir da transformada de Brill, validaram a inclusão desse algoritmo nos serviços de etiquetagem morfossintática oferecidos pelo PortService-Br.

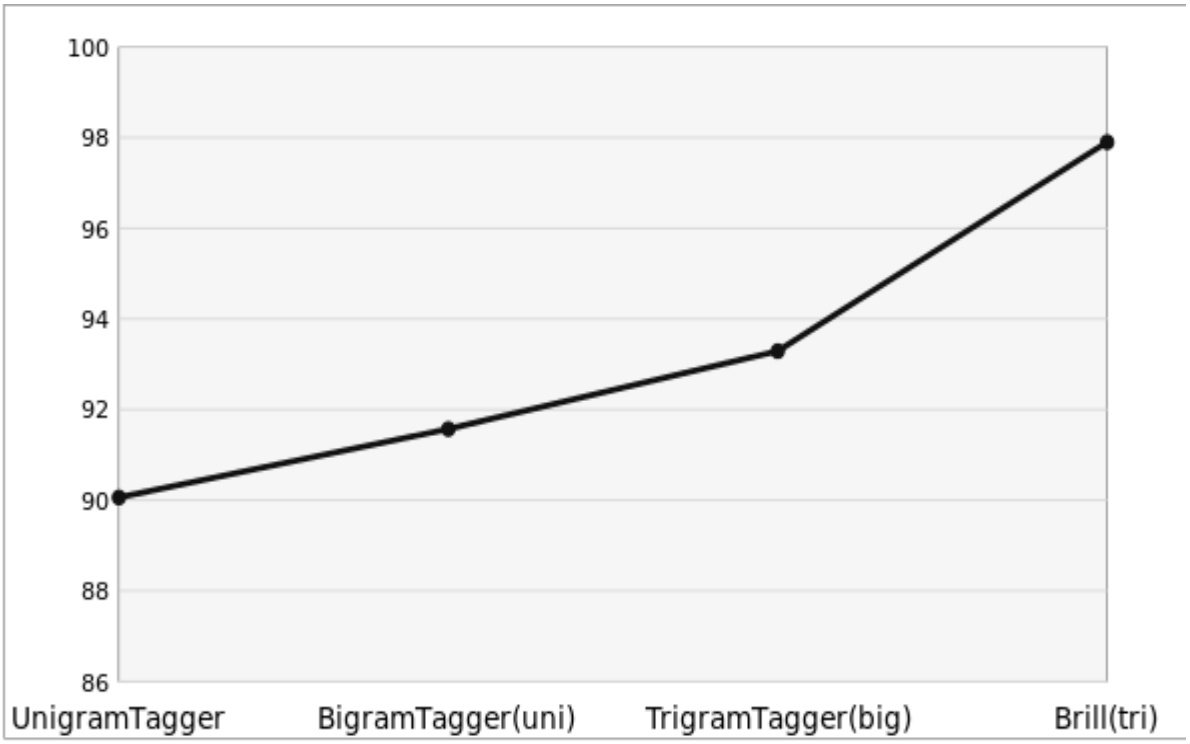

Figura 18. Comparação da acurácia de cada algoritmo.

Vale ressaltar que para alcançar um bom resultado, foi preciso investir um tempo considerável de processamento computacional e uma forte pesquisa em algoritmos que fossem capazes de processar a grande quantidade de dados gerados no experimento. Também é preciso um grande conhecimento em programação para realizar tais experimentos. Pesquisadores em PLN geralmente não possuem todos os conhecimentos necessários para a realização de experimentos dessa magnitude.

Esse fato reforça a viabilidade de se disponibilizar esses algoritmos em uma plataforma centralizada com livre acesso a pesquisadores.

5.3 Implementação dos serviços do PortService-Br

Com a intenção de disponibilizar serviços remotos e de modo a tornar a implementação rápida e ao mesmo tempo eficiente, foi preciso descobrir um *framework* que facilitasse todo esse processo. Como relatado anteriormente, a linguagem de programação usada no projeto é o Python. Por isso decidiu-se optar pelo uso do *Django RestFramework*.

Django REST Framework é um dos mais conhecidos frameworks para implementação de APIs do tipo REST em conjunto com o Django. Destaca-se por ser muito leve e por disponibilizar uma API que permite abstrair imensos passos criando um serviço eficiente em termos de implementação e de uso.

Alguns dos serviços disponibilizados na plataforma consomem muitos recursos dos sistemas hospedeiros e sua inicialização nos servidores torna-se bastante lenta. De modo a tornar as requisições mais eficientes em termos de velocidade de resposta, foi necessário desenvolver um algoritmo no servidor que fosse capaz de iniciar os recursos em conjunto com a inicialização do servidor e os armazenasse em memória.

O algoritmo do servidor consiste, em primeiro lugar, de carregar os módulos na memória, e em segundo na espera por requisições aos serviços que são fornecidos via HTTP GET. Quando recebe algum pedido, vai então verificar se os parâmetros estão corretos e qual serviço deve ser executado.

As próximas seções apresentam em detalhes o desenvolvimento de cada um dos serviços oferecidos na plataforma PortService-Br, a saber: Análise léxico semântica, etiquetagem morfossintática e segmentação de textos em frases e palavras.

5.3.1 Implementação dos recursos de análise léxico semântica

A implementação deste recurso demandou bastante esforço em termos de programação. Para disponibilizar esse recurso foi necessário o uso de três WNs da língua portuguesa: a OpenWordnet-PT, a ONTO.PT e a PULO.

A OpenWordnet-PT e a ONTO.PT se encontram em arquivo no formato RDF. Para que fosse possível seu uso tornou-se necessário pesquisar bibliotecas para a linguagem Python com a finalidade de leitura e manipulação de arquivos RDF. Foi encontrada a biblioteca RDFLib, um toolkit gráfico baseado em Python, que possui interface para consultas baseadas na linguagem de consulta SPARQL. O retorno de consultas em SPARQL geralmente são expressas em forma de triplas (*subject, predicat, object*). A WN PULO é uma base de dados relacional MySQL, seu uso foi um pouco mais simples bastando apenas a realização de consultas em SQL.

Para possibilitar a consulta de dados representados em RDF, utiliza-se a linguagem de consulta SPARQL, que seria o equivalente a linguagem SQL para os bancos de dados relacionais. SPARQL é um conjunto de especificações que fornecem linguagens e protocolos para consultar e manipular o conteúdo gráfico RDF.

SPARQL possibilita que uma Query possa ser usada para formular consultas que variam de padrão gráfico simples de correspondência até consultas complexas envolvendo diversos formatos de RDF distribuídos na Web (Gabriel Junior, 2016). O retorno das consultas do serviço de análise léxico semântica é um arquivo no formato JSON, cuja estrutura é apresentada no Quadro 2.

Quadro 2. Estrutura do retorno JSON do serviço de análise léxico semântica.

```
[
    {
        "palavra": "",
        "synsets": [
            {
                "type": "",
                "gloss": "",
                "synsetId": "",
                "formasLexicais": []
            }
        ]
    }
]
```

A estrutura consiste em uma lista principal onde cada “palavra” constitui um item da lista, cada item contém uma lista de “synsets” com os seguintes dados: O primeiro dado retornado na consulta “synsetId” se refere ao identificador único que identifica cada synset. O segundo “synset” se refere a URL da instância. O terceiro “gloss” é referente a glosa que pode ou não conter uma definição do synset. O quarto “type” se refere à classe gramatical do synset. E por fim o quinto “formasLexicais” se refere a todas as palavras associadas com o “synsetId”, ou seja, representam o mesmo sentido.

As próximas seções apresentam em detalhes o desenvolvimento dos serviços de análise léxico semântica, com cada uma das WNs disponibilizadas no projeto.

5.3.1.1 Implementação com a OpenWordNet-PT

A OpenWordNet-PT está disponível como um arquivo RDF. Desta forma sua leitura é realizada com auxílio da biblioteca RDFLib. Inicialmente foi preciso baixar o recurso no site do projeto OpenWordNet-PT, versão 3.0 no formato RDF/XML. A versão 3.0 do recurso conta, até o momento, com: 43.925 synsets, dos quais 32.696 correspondem a substantivos, 4.675 a verbos, 5.575 a adjetivos e 979 a advérbios.

O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o OPenWOrdNet-PT possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser analisada pela WN.

A seguir é apresentado um exemplo de URL de consulta passando como parâmetro a palavra “computador”. No exemplo foi usado somente uma palavra para facilitar a apresentação do serviço, mas textos maiores também podem ser usados no parâmetro.

http://portservice.pythonanywhere.com/analise/lexicosemantica/openwordnet/?texto=computador

Ao ser recebido pelo servidor uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo AnaliseLexicoSemantica. Neste módulo primeiramente o texto é dividido em uma lista de palavras (no exemplo acima essa fase não é

necessária) para a divisão do texto em palavras é usado *word_tokenize* da biblioteca NLTK, passando como parâmetro de divisão descritores da língua portuguesa.

A lista de palavras é passada como parâmetro de entrada para o módulo EtiquetagemMorfossintatica. A etiquetagem se faz necessária para se obter os lemas de cada palavra e sua distribuição morfossintática. A WN OpenWordNet-PT somente armazena palavra em sua forma primitiva, ou seja, lematizadas. Dessa forma para poder ser consultada, uma palavra precisa estar lematizada.

OpenWordNet-PT também só armazena verbos, advérbios, substantivos e adjetivos, dessa forma se faz necessário verificar se a palavra se encaixa em ums das quatro classes de palavras antes da consulta. O módulo de EtiquetagemMorfossintatica então gera uma lista de palavras etiquetadas e lematizadas.

Cada palavra da lista de palavras etiquetadas é então repassada para a WN selecionada. Neste caso o módulo que realiza a análise usa a OpenWordNet-PT. É então gerada uma lista de objetos da classe ALSDado para cada palavra contida no texto de entrada.

O Quadro 3 apresenta o *script* de consulta em SPARQL. Esse script usa a biblioteca RDFLib para pesquisar synsets relativos a palavra “computador” na OpenWordnet-PT.

Quadro 3. Consulta em SPARQL na OpenWordNet-PT usando a biblioteca RDFLib para pesquisar synsets relativos a palavra “computador”.

```
select ?synsetId ?synset ?type ?gloss
   {
      ?w <https://w3id.org/own-pt/wn30/schema/lexicalForm> "computador@pt .
      ?ws <https://w3id.org/own-pt/wn30/schema/word> ?w .
      ?synset <https://w3id.org/own-pt/wn30/schema/containsWordSense> ?ws
.
      ?synset <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type .
      ?synset <https://w3id.org/own-pt/wn30/schema/synsetId> ?synsetId
      optional { ?synset <https://w3id.org/own-pt/wn30/schema/gloss> ?gloss .}
```

```
}
```

O retorno desta consulta acima é uma tripla contendo as informações sobre a palavra passada como objeto de pesquisa. A requisição retorna uma lista de *synsets* relativos a cada synset encontrado para a palavra “computador”. No Quadro 4 é apresentado o retorno da requisição no formato JSON.

Quadro 4. Retorno da requisição para a palavra “computador”.


```
[
    {
      "palavra": "computador",
      "synsets": [
        {
          "type": "NounSynset",
          "synset": "https://w3id.org/own-pt/wn30-pt/instances/synset-03082979-n",
          "gloss": "",
           "synsetId": "3082979"
        },
        {
          "type": "NounSynset",
          "synset": "https://w3id.org/own-pt/wn30-pt/instances/synset-02938886-n",
           "gloss": "",
           "synsetId": "2938886"
        },
        {
          "type": "NounSynset",
          "synset": "https://w3id.org/own-pt/wn30-pt/instances/synset-09887034-n",
          "gloss": "",
           "synsetId": "9887034"
        }
       ]

       }
  ]
```


Para simplificar a visualização somente é mostrado um *synset*, porém a palavra “computador” contém outros *synsets* associados a ela.

5.3.1.2 Implementação com a ONTO.PT

A ONTO.PT está disponível como um arquivo RDF. Dessa forma sua leitura, assim como a OpenWordNet-PT, também é realizada com auxílio da biblioteca RDFLib.

Inicialmente foi preciso baixar o recurso no site do projeto, foi baixada a versão mais recente do recurso 0.6 no formato RDF/XML. A versão 0.6 do recurso conta com 117.000 synsets, divididos da seguinte forma: 67.873 substantivos, 26.451 verbos, 20.760 adjetivos e 2.366 advérbios. O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço.

A requisição aos serviços usando a ONTO.PT possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser analisada pela WN.

A seguir é apresentado um exemplo de consulta passando como parâmetro a palavra “programar”. No exemplo foi usado somente uma palavra para facilitar a apresentação do serviço. Porém textos maiores também podem ser usados como parâmetro.

http://portservice.pythonanywhere.com/analise/lexicosemantica/ontopt/?texto=programar

Ao ser recebido pelo servidor uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo AnaliseLexicoSemantica. Neste módulo primeiramente o texto é dividido em uma lista de palavras (no exemplo acima essa fase não é necessária) para a divisão do texto em palavras é usado *word_tokenize* da biblioteca NLTK passando como parâmetro de divisão descritores da língua portuguesa.

A lista de palavras é passada como parâmetro de entrada para o módulo EtiquetagemMorfossintatica. A etiquetagem se faz necessária para se obter os lemas de cada palavra e sua distribuição morfossintática. Como na OpenWordNet-PT o ONTO.PT somente armazena palavras em sua forma primitiva, ou seja, lematizadas. Dessa forma para poder consultar uma palavra a mesma precisa estar lematizada. A ONTO.PT também somente armazena verbos, advérbios, substantivos e adjetivos, dessa forma se faz necessário verificar se a palavra se encaixa em uma dessas quatro classes, antes da consulta. O módulo de EtiquetagemMorfossintatica

então gera uma lista de palavras etiquetadas e lematizadas.

Cada palavra da lista de palavras etiquetadas é então repassada para a WN selecionada. Neste caso o módulo realiza a análise usando a ONTO.PT. É então gerada uma lista de objetos da classe ALSDado para cada palavra contida no texto de entrada. O Quadro 5 apresenta o script de consulta em SPARQL utilizando a biblioteca RDFLib para pesquisar synsets relativos à palavra “programar”, na ONTO.PT.

Quadro 5. Consulta em SPQRQL no ONTO.PT usando a biblioteca RDFLib para pesquisar conceitos relativos a palavra “programar”.

```
select ?synsetId ?synset ?type ?gloss
  {
     ?w <http://ontopt.dei.uc.pt/OntoPT.owl#formaLexical> “programar”.
      ?synset <http://ontopt.dei.uc.pt/OntoPT.owl#definicao> ?w .
      ?synset <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type .
      ?synset <http://www.w3.org/2000/01/rdf-schema#label> ?synsetId
       optional { ?synset <http://ontopt.dei.uc.pt/OntoPT.owl#definicao> ?gloss .}
  }
```

A requisição retorna uma lista de synsets relativos a palavra “programar”. o Quadro 6 apresentado o retorno da requisição no formato JSON.

Quadro 6. Retorno da requisição para a palavra “programar”.

```
[
  {
     "palavra": "programar",
     "synsets": [
        {
           "type": "VerboSynset",
           "synsetId": "89986",
```

```
        "gloss": "--",
        "formasLexicais": [
            "condicionar",
            "programar"
        ]
    },
    {
        "type": "VerboSynset",
        "synsetId": "90422",
        "gloss": "--",
        "formasLexicais": [
            "planificar",
            "programar",
            "agendar"
        ]
    },
    {
        "type": "VerboSynset",
        "synsetId": "90754",
        "gloss": "--",
        "formasLexicais": [
            "planear",
            "planejar",
            "programatizar",
            "programar",
            "calendarizar"
```

```
            ]
          }
        ]
      }
]
```

Para simplificar a visualização somente é mostrado um *synset*, porém a palavra gostar contém outros *synsets* associados a ela.

5.3.1.3 Implementação com PULO

O PULO está disponível como uma base de dados SQL, em um banco de dados MYSQL. Dessa forma sua leitura e manipulação são realizadas mediante consultas na linguagem SQL. Inicialmente foi preciso baixar a base de dados do recurso na página do projeto PULO, foi baixada a versão mais recente da base 3.0. A Figura 19 apresenta o diagrama da base de dados do PULO.

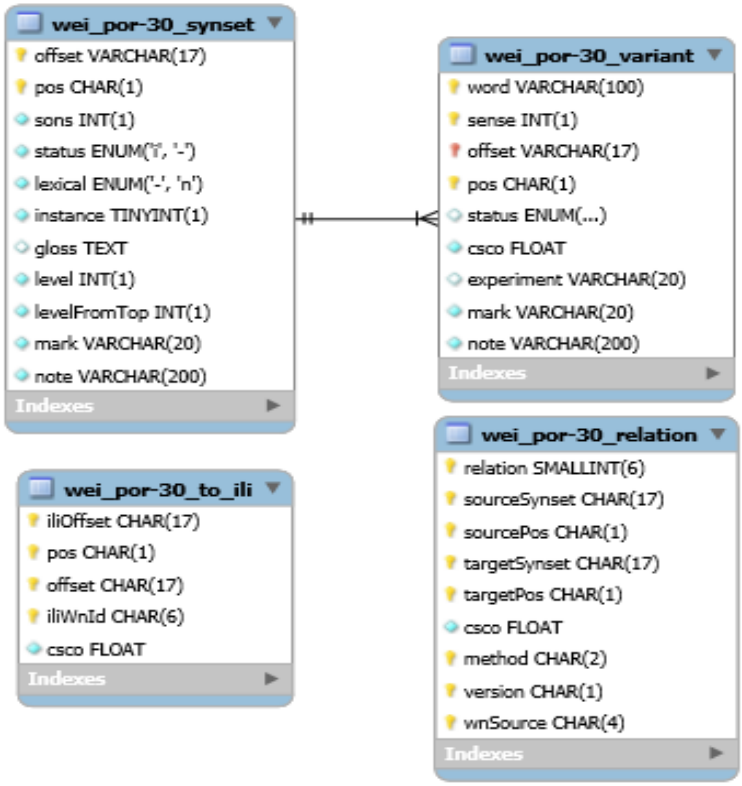

Figura 19. Diagrama da base de dados do PULO.

Posteriormente foi necessário a exportação de toda a base para o banco de dados do servidor. A versão 3.0 do recurso conta com 17.942 synsets, divididos da seguinte forma: 3.581 adjetivos, 528 advérbios, 3.786 verbos e 10.047 substantivos.

O acesso ao recurso está disponível via requisição HTTP GET, bastando apenas a realização de uma requisição para a URL do serviço. A requisição ao serviço usando o PULO possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser analisada pela WN.

A seguir é apresentada um exemplo de URL de consulta passando como parâmetro a palavra “claramente”. No exemplo foi usado somente uma palavra para facilitar a apresentação do serviço, porém, textos maiores também podem ser usados no parâmetro.

http://portservice.pythonanywhere.com/analise/lexicosemantica/pulo/?texto=claramente

Ao ser recebido pelo servidor uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo AnaliseLexicoSemantica. Nesse módulo primeiramente o texto é dividido em uma lista de palavras (no exemplo acima essa fase não é necessária) para a divisão do texto em palavras é usado *word_tokenize* da biblioteca NLTK, passando como parâmetro de divisão descritores da língua portuguesa.

A lista de palavras é passada como parâmetro de entrada para o módulo EtiquetagemMorfossintatica. A etiquetagem se faz necessária para se obter os lemas de cada palavra e sua distribuição morfossintática. A WN PULO somente armazena palavra em sua forma primitiva, ou seja, lematizadas. Dessa forma para poder ser consultada, uma palavra precisa estar lematizada.

PULO também só armazena verbos, advérbios, substantivos e adjetivos. Dessa forma se faz necessário verificar se a palavra se encaixa em uma das quatro classes, antes da consulta. O módulo de EtiquetagemMorfossintatica então gera uma lista de palavras etiquetadas e lematizadas.

Cada palavra da lista de palavras etiquetadas é então repassada como parâmetro de consulta, nesse caso o módulo que realiza a análise usando PULO. É então

gerada uma lista de objetos da classe ALSDado para cada palavra contida no texto de entrada. O Quadro 7 apresenta o script de consulta SQL para pesquisar synsets relativos a palavra “claramente”.

Quadro 7. Consulta em SQL na base de dados PULO para pesquisar synsets relativos a palavra “claramente”.

```
select  s.pos, s.gloss,s.offset from  wei_por_30_variant v , wei_por_30_synset s
where v.word= ‘claramente’ and v.offset = s.offset
```

Para se obter todas as formas lexicais referentes a cada synset, foi preciso a realização de uma nova consulta ao banco de dados PULO, passando como parâmetro cada synsetId retornado na consulta anterior. O Quadro 8 apresenta o script de consulta das formas lexicais por synset.

Quadro 8. Script de consulta no PULO de formas lexicais por synset.

```
select  v.word from  wei_por_30_variant v , wei_por_30_synset s where s.offset=
‘id’ and v.offset = s.offset
```

A requisição retorna uma lista de formas lexicais relativos a cada synset encontrado para a palavra “claramente”. O Quadro 9 apresenta o retorno da requisição no formato JSON.

Quadro 9. Retorno da consulta para a palavra “claramente”.


```
[
    {
        "palavra": "claramente"
        "synsets": [
            {
                "type": "r"
                "gloss": " frequentemente usado informalmente para abertamente",
                "synsetId": "por-30-00039318-r",
                "formasLexicais": [
                    "abertamente",
```

```
                "aparentemente",
                "certamente",
                "claramente",
                "claro",
                "com_certeza",
                "declaradamente",
                "evidentemente",
                "explicitamente",
                "francamente",
                "indiscutivelmente",
                "indubitavelmente",
                "inegavelmente",
                "inquestionavelmente",
                "manifestamente",
                "marcadamente",
                "naturalmente",
                "notavelmente",
            ],
        },

    }
]
```

Para simplificar a visualização somente é mostrado um *synset*, porém a palavra “claramente” contém outros *synsets* associados a ela.

5.3.2 Implementação do serviço de etiquetagem morfossintática

Para a implementação do serviço de etiquetagem morfossintática foram usados três etiquetadores. Dois deles, já consagrados e amplamente utilizados por pesquisadores de PLN como serviços locais, são o Freeling e o TreeTagger. O Terceiro etiquetador foi desenvolvido no âmbito deste projeto intitulado etiquetador Padrão. As ferramentas Freeling e TreeTagger necessitam de prévia instalação e configuração na máquina servidora e necessitam de muitas dependências para seu pleno funcionamento.

O retorno das consultas do serviço de etiquetagem morfossintática usando TreeTagger e o Freeling é um arquivo no formato JSON, cuja estrutura é apresentada no Quadro 10.

Quadro 10. Estrutura do retorno JSON do serviço de etiquetagem morfossintática usando Freeling e TreeTagger.

```
[
  {
    "lemma": "",
    "token": "",
    "poss": “"
  }
]
```

A estrutura consiste em uma lista principal, onde cada token to texto de entrada representa um item. A propriedade “lemma” é referente ao lema da palavra, ou seja, sua forma primitiva. A propriedade “token” é referente aos elementos de cada oração, que podem ser palavras, números e sinais de pontuação. A propriedade “poss” é referente à distribuição morfossintática.

O retorno das consultas do serviço de etiquetagem morfossintática usando o Etiquetador Padrão é um arquivo no formato JSON, cuja estrutura é apresentada no Quadro 11.

Quadro 11. Estrutura do retorno JSON do serviço de etiquetagem morfossintática usando o etiquetador Padrão.

```
[
    {
        "tempo": "",
        "genero": "",
        "token": "",
        "poss": "",
        "pessoa": "",
        "numero": "",
        "lemma": "",
        "subclasse": "",
        "modo": ""
    }

]
```

A estrutura apresenta as mesmas propriedades do arquivo JSON dos etiquetadores TreeTagger e Freeling e agrega algumas propriedades específicas do etiquetador Padrão: A propriedade “genero” é referente ao gênero do token e aplica-se a substantivos, adjetivos e determinantes. A propriedade “tempo” é referente a verbos, específica o tempo verbal. A propriedade “pessoa” referente a verbos determina a pessoa da conjugação. A propriedade “numero” é referente a substantivos, adjetivos e determinantes, podem estar no plural ou singular. A propriedade “subclasse” é referente a determinantes. A propriedade “modo” referente a verbos, determina o modo verbal.

As próximas seções trazem mais detalhes da implementação do serviço de etiquetagem morfossintática com os três etiquetadores disponíveis.

5.3.2.1 Implementação usando Freeling

A implementação do serviço de etiquetagem morfossintática usando Freeling demandou bastante esforço tanto a nível de implementação quanto de instalação e configuração da ferramenta no sistema.

Inicialmente foi preciso baixar o recurso no site do projeto do Feeling. Além do etiquetador, muitas outras dependências precisam ser instaladas no ambiente de desenvolvimento para pleno funcionamento. Freeling requer pelo menos 3GB de espaço livre em disco. Além disso para a instalação em ambientes Linux e MacOSX as seguintes dependências são necessárias:

1. libicu.
2. libboost-regex.
3. libboost-system.
4. libboost-thread.
5. libboost-program-options.
6. libboost-locale (somente requerida para MacOSX ou FreeBSD, não requerida em Linux).

O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o Freeling possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser etiquetada.

Abaixo é apresentado um exemplo de URL de consulta passando como parâmetro a frase “Um belo dia para programar.”.

http://portservice.pythonanywhere.com/analise/morfossintatica/freeling/?texto=Um belo dia para programar.

Ao ser recebida uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo de EtiquetagemMorfossintatica. O módulo é responsável por executar o método que faz a etiquetagem com o etiquetador Freeling.

Após sua instalação e configuração, *Freeling* somente pode ser executado via linha de comandos, sendo assim tornou-se necessário a implementação de algoritmos capazes de fazer chamadas a funcionalidades do sistema operacional do servidor, para acionar as funções do *Freeling*. Felizmente Python conta com classes nativas para essa tarefa, dessa forma tornando-as mais eficiente. O Quadro 12 apresenta o método de acesso ao Freeling instalado no servidor usando Python.

Quadro 12. Método de acesso ao Freeling instalado no servidor usando Python.

```
def EtiquetagemFreeeling(self,texto):
        listaEtiquetada = list()
        processo = Popen(['analyze -f pt.cfg'],stdin=PIPE, stdout=PIPE,shell=True)
         saida = str(processo.communicate(bytes (texto, 'UTF-8'))[0])
          listaSaida = saida[2:len(saida)].split('\\n')
                    for item in listaSaida [0:len(listaSaida )-2]:
                        lista = item.split(' ')
                        eMDado= EMDado()
                       eMDado.Palavra = str(lista[0]).replace('n',"")
```

```
                    eMDado.Lemma = str(lista[1])
                    eMDado.Poss = str(lista[2])
                   listaEtiquetada .append(eMDado)
        return listaEtiquetada
```

O texto recebido pelo método EtiquetagemFreeeling é passado como parâmetro para uma chamada de sistema ao etiquetador Freeling instalado no servidor. A saída desse processo é em forma de texto plano, dessa forma devendo ser convertido para uma lista. A lista é percorrida e as informações são armazenadas em um objeto do tipo EMDado. Cada objeto criado é adicionado à lista de tokens etiquetados. Ao final se tem uma lista com todos os tokens do texto etiquetados e prontos para serem retornados como resposta a requisição. O Quadro 13 apresenta o retorno da requisição ao serviço de etiquetagem morfossintática usando Freeling, para a frase “Um belo dia para programar.”

Quadro 13. Retorno da requisição ao serviço de etiquetagem com Freeling para a frase “Um belo dia para programar.”


```
[
    {
        "poss": "DI0",
        "lemma": "um",
        "token": "Um"
    },
    {
        "poss": "AQ0",
        "lemma": "belo",
        "token": "belo"
    },
    {
        "poss": "NCMS",
        "lemma": "dia",
        "token": "dia"
    },
    {
        "poss": "SPS",
```

```
        "lemma": "para",
        "token": "para"
    },
    {
        "poss": "VMN",
        "lemma": "programar",
        "token": "programar"
    },
    {
        "poss": "Fp",
        "lemma": ".",
        "token": "."
    }
]               "token": "programar",
                "poss": "VMN"
            }
        },
        {
            "token": {
                "lemma": ".",
                "token": ".",
                "poss": "Fp"
            }
        }
    ]
}
```


Para maiores informações sobre o Freeling e seu conjunto de etiquetas é recomendado visitar o site oficial do etiquetador, onde é possível encontrar todo o conjunto de recursos disponível em http://nlp.lsi.upc.edu/freeling/.

5.3.2.2 Implementação usando Treetagger

Assim como a implementação do serviço de etiquetagem morfossintática usando Freeling, a implementação usando o Treetagger demandou bastante esforço tanto a nível de implementação quanto de instalação e configuração da ferramenta no sistema.

Inicialmente foi preciso baixar o projeto do TreeTagger e os parâmetros para se trabalhar com a língua portuguesa. Além do etiquetador e seus parâmetros, muitas

outras dependências precisam ser instaladas no ambiente de desenvolvimento para pleno funcionamento. Os passos para instalação são os seguintes:

1. Baixar o pacote de instalação para um dos sistemas (PC-Linux, Mac OS-X, ARM-Linux).
2. Baixar os scripts instalação do etiquetador no mesmo diretório do pacote de instalação.
3. Baixar os scripts instalação “install-tagger.sh”.
4. Baixar os parâmetros de linguagem
5. Abrir um terminal no diretório de instalação e executar os scripts “sh install-tagger.sh”

Após sua instalação e configuração, o TreeTagger, diferentemente de Freeling, pode ser armazenado em um objeto em memória no servidor sendo carregado em sua inicialização. O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o TreeTagger possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser etiquetada.

Abaixo é apresentado um exemplo de URL de consulta passando como parâmetro a frase “É um projeto fácil.”.

http://portservice.pythonanywhere.com/analise/morfossintatica/treetagger/?texto=É um projeto fácil.

Ao ser recebida uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo de EtiquetagemMorfossintatica. O módulo é responsável por executar o método que fará a etiquetagem com o etiquetador TreeTagger. O Quadro 14 apresenta o método responsável por acessar o TreeTagger.

Quadro 14. Método de etiquetagem usando TreeTagger.

```
def EtiquetagemTreeTagger(self,texto):
```

```
listaEtiquetada = list()
lista = TreeTagger.tag(texto)

for sent in lista:
        eMDado= EMDado()
        eMDado.Palavra  = sent[0]
        eMDado.Possl = sent[1]
         eMDado.Lemma = sent[2]
        listaEtiquetada .append(eMDado)
return listaEtiquetada
```

O texto é recebido pelo método EtiquetagemTreeTagger (o etiquetador TreeTagger já se encontra carregado em memória) e gera uma lista com a etiquetagem. A lista é percorrida e as informações são armazenadas em um objeto do tipo EMDado e cada objeto criado é adicionado à lista de tokens etiquetados. Ao final, tem-se uma lista com todos os tokens do texto etiquetados e prontos para serem retornados como resposta a requisição. O Quadro 15 apresenta o retorno da requisição ao serviço de etiquetagem morfossintática com TreeTagger para a frase “Programadores são ferramentas para converter cafeína em código.

Quadro 15. Retorno da requisição ao serviço de etiquetagem com TreeTagger para a frase “Programadores são ferramentas para converter cafeína em código.”


```
[
   {
      "lemma": "programador",
      "poss": "NCMP",
      "token": "Programadores"
   },
```

```
{
    "lemma": "ser",
    "poss": "VMI",
    "token": "são "
},
{
    "lemma": "ferramenta",
    "poss": "NCFP",
    "token": "ferramentas"
},
{
    "lemma": "para",
    "poss": "SPS",
    "token": "para"
},
{
    "lemma": "converter",
    "poss": "VMN",
    "token": "converter"
},
{
    "lemma": "cafe\u00edna",
    "poss": "NCFS",
    "token": "cafe\u00edna"
},
{
```

```
        "lemma": "em",
        "poss": "SPS",
        "token": "em"
    },
    {
        "lemma": "código ",
        "poss": "NCMS",
        "token": "código "
    },
    {
        "lemma": ".",
        "poss": "Fp",
        "token": "."
    }
]
```


Para maiores informações sobre o TreeTagger e seu conjunto de etiquetas é recomendado visitar o site oficial do etiquetador disponível em http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/.

5.3.2.3 Implementação usando etiquetador padrão

A implementação do serviço de etiquetagem morfossintática usando o etiquetador Padrão, desenvolvido no âmbito deste trabalho, foi um pouco mais simples de disponibilizar.

O etiquetador padrão, assim como o TreeTagger, pode ser armazenado em um objeto em memória no servidor sendo carregado em sua inicialização. O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o etiquetador Padrão possui o

seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser etiquetada.

Abaixo é apresentado um exemplo de URL de consulta passando como parâmetro a frase “Programador feliz”.

http://portservice.pythonanywhere.com/analise/morfossintatica/padrao/?texto=Programador feliz.

Ao ser recebida uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo de EtiquetagemMorfossintatica. O módulo é responsável por executar o método que faz a etiquetagem com o etiquetador Padrão. O Quadro 16 apresenta o método responsável por acessar o etiquetador Padrão.

Quadro 16. Método de etiquetagem usando etiquetador Padrão.

```
def EtiquetagemPadrao(self, texto):
        listaEtiquetada = list()
        lista= Padrao.tag(texto)

        if lista!= None:
            for item in lista:
                if item[1]!= None:
                  eMDadoPadrao = self.GetDatabase(item[0],item[1])
                 listaTokensClassificados.append(token)

        return listaEtiquetada
```

O texto é recebido pelo método EtiquetagemPadrao, dado como entrada para o etiquetador Padrão, carregado em memória. Esse método cria uma lista com a

etiquetagem gerada. A lista é percorrida e as informações são armazenadas em um objeto do tipo EMDadoPadrao e cada objeto criado é adicionado à lista de tokens etiquetados. Ao final, tem-se uma lista com todos os tokens do texto etiquetados e prontos para serem retornados como resposta a requisição. O Quadro 17 apresenta o retorno da requisição ao serviço de etiquetagem morfossintática com o etiquetador Padrão para a frase “Programador feliz”.

Quadro 17. Retorno da requisição ao serviço de etiquetagem com o etiquetador Padrão para a frase “Programador feliz”.


```
[
    {
        "poss": "N",
        "subclasse": "",
        "numero": "S",
        "genero": "M",
        "token": "programador",
        "lemma": "programador",
        "tempo": "",
        "modo": "",
        "pessoa": ""
    },
    {
        "poss": "ADJ",
        "subclasse": "",
        "numero": "S",
        "genero": "M/F",
        "token": "feliz",
        "lemma": "feliz",
        "tempo": "",
        "modo": "",
        "pessoa": ""
    }
]
```


Para maiores informações sobre o etiquetador Padrão e seu conjunto de etiquetas é recomendado visitar o site oficial do etiquetador, disponível em http://portservice.pythonanywhere.com.

5.3.3 Implementação dos serviços de segmentação de textos

Para a implementação dos recursos de segmentação de textos, foram usados recursos nativos da biblioteca NLTK. Para a segmentação de textos em frases foi usado o *sent_tokenize*, com parâmetros descritores para a língua portuguesa. Para a segmentação de textos em palavras, foi usado o *word_tokenize* com parâmetros descritores para a língua portuguesa. O retorno da consulta ao serviço de segmentação de textos em sentenças é um arquivo no formato json cuja estrutura é apresentada no quadro 18.

Quadro 18. Estrutura do retorno JSON do serviço segmentação de textos em sentenças.

```
{
    [
      {
          "sentenca": ""

      },
    ]
}
```

A estrutura consiste em uma lista principal onde cada sentença representa um item. A propriedade “sentenca” é referente a cada sentença encontra no texto de entrada. A segmentação de textos em sentenças é aplicada por meio da identificação de caracteres finalizadores de sentenças, principalmente, dos sinais de pontuação. Embora se use o ponto final (ou ponto de interrogação ou exclamação) na maioria das línguas para indicar o fim de frase, nem sempre isso acontece. O retorno da consulta ao serviço de segmentação de textos em palavras é um arquivo no formato json sua estrutura é apresentada no Quadro 19.

Quadro 19. Estrutura do retorno JSON do serviço segmentação de textos em palavras.

```
{
  [
      {
          "token": ""

      },
```

```
    ]
}
```

A estrutura consiste em uma lista principal, onde cada token representa um item. A propriedade “token” é referente a cada palavra, número e sinais de pontuação encontrado no texto de entrada. Essa tarefa utiliza, basicamente os sinais gráficos, tais como espaços, e algoritmos para o reconhecimento de entidades limítrofes de um *token*. As próximas seções trazem mais detalhes da implementação do serviço de segmentação de textos disponíveis.

5.3.3.1 Implementação do serviço de segmentação de texto em sentenças

A implementação do serviço de segmentação de textos em sentenças, foi uma tarefa relativamente simples, bastando apenas a importação do pacote *sent_tokenize* da biblioteca NLTK. O acesso ao serviço está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o *sent_tokenize* possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser etiquetada.

Abaixo é apresentado um exemplo de URL de consulta passando como parâmetro o texto “Qualquer um pode escrever um código que o computador entenda. Bons programadores escrevem códigos que os humanos entendam.”

http://portservice.pythonanywhere.com/analise/segmentacao/sentenca/?texto=Qualquer um pode escrever um código que o computador entenda. Bons programadores escrevem códigos que os humanos entendam.

Ao ser recebida uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo de Segmentacao. Esse módulo é responsável por executar o método que faz a segmentação do texto em sentenças. O Quadro 20 apresenta o método responsável por segmentar um texto em frases.

Quadro 20. Método responsável por segmentar um texto em frases.

```
def Segmentar(self,texto):
    sentencas = sent_tokenize(texto,language = 'portuguese')
    return sentencas
```

O texto é recebido pelo método Segmentar da classe SegmentacaoSentenca e passado como parâmetro para o *sent_tokenize,* que gera uma lista contendo todas as sentenças, do texto passado como parâmetro, prontas para serem retornadas como resposta a requisição. O Quadro 21 apresenta o retorno da requisição para o texto passado como exemplo acima.

Quadro 21. Retorno da requisição ao serviço de segmentação de texto em sentenças para a texto de exemplo.

```
[
  {
    "frase": "Qualquer um pode escrever um código  que o computador entenda"
  },
  {
    "frase": "Bons programadores escrevem códigos que os humanos entendam"
  }
]
```

Para maiores informações sobre o *sent_tokenize* é recomendado visitar o site oficial do NLTK, onde é possível encontrar todo o conjunto de recursos disponível em http://www.nltk.org/.

5.3.3.2 Implementação do serviço de segmentação de texto em palavras

A implementação do serviço de segmentação de textos em palavras, foi uma tarefa relativamente simples, bastando apenas a importação do pacote *word_tokenize* da biblioteca NLTK. O acesso ao serviço está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço usando o *word_tokenize* possui o seguinte parâmetro:

- **texto**: Especifica o texto ou palavra a ser etiquetada.

Abaixo é apresentado um exemplo de URL de consulta passando como parâmetro o texto “Qualquer um pode escrever um código que o computador entenda. Bons programadores escrevem códigos que os humanos entendam.”

http://portservice.pythonanywhere.com/analise/segmentacao/palavra/?texto=Qualquer um pode escrever um código que o computador entenda. Bons programadores escrevem códigos que os humanos entendam.

Ao ser recebida uma requisição, o conteúdo do parâmetro “texto” é repassado para o módulo de Segmentacao. O módulo é responsável por executar o método que faz a segmentação do texto em palavras. O Quadro 22 apresenta o método responsável por segmentar um texto em palavras.

Quadro 22. Método responsável por segmentar um texto em palavras.

```
def Segmentar(self,texto):
    lista = word_tokenize(texto,language='portuguese')

    return lista
```

O texto é recebido pelo método Segmentar, da classe SegmentacaoPalavra, e passado como parâmetro para o *word_tokenize,* que gera uma lista contendo todas as palavras ou tokens do texto passado como parâmetro, prontas para serem retornadas como resposta a requisição. O Quadro 23 apresenta o retorno da requisição para o texto passado como exemplo acima.

Quadro 23. Retorno da requisição ao serviço de segmentação de texto em palavras para o texto de exemplo.

```
[
    {
        "token": "qualquer"
    },
    {
        "token": "um"
    },
    {
```

```
    "token": "pode"
},
{
    "token": "escrever"
},
{
    "token": "um"
},
{
    "token": "código"
},
{
    "token": "que"
},
{
    "token": "o"
},
{
    "token": "computador"
},
{
    "token": "entenda"
},
{
    "token": "."
},
{
    "token": "bons"
},
{
    "token": "programadores"
},
{
    "token": "escrevem"
},
{
    "token": "códigos"
},
{
    "token": "que"
},
{
    "token": "os"
```

```
    },
    {
        "token": "humanos"
    },
    {
        "token": "entendam"
    },
    {
        "token": "."
    }
]
```


Para maiores informações sobre o *word_tokenize* é recomendado visitar o site oficial do NLTK, onde é possível encontrar todo o conjunto de recursos disponível em http://www.nltk.org/.

5.4 Interface de consulta Web

Para além da API de resposta a pedidos REST com componentes essenciais implementados, a plataforma PortService-Br também oferece uma interface web que permite ao utilizador a realização de consultas via interface visual, para consultar e usar os serviços oferecidos, os parâmetros utilizados em cada serviço e a descrição do projeto. A Figura 20 apresenta a interface web de consulta aos serviços do PortService-Br.

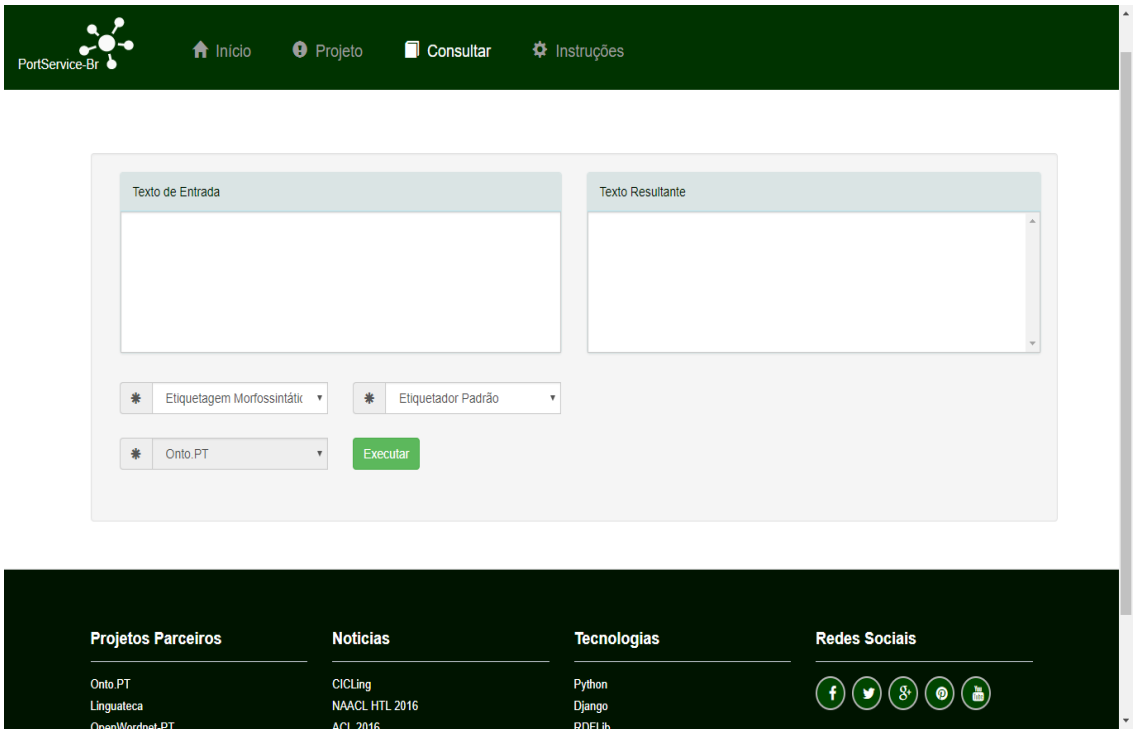

Figura 20. Interface web de consulta aos serviços do PortService-Br.

Como pode ser visualizado na Figura 20 a interface de consulta web é simples e intuitiva. Contém uma caixa de entrada de texto e o usuário pode escolher qual serviço quer testar. Se a escolha for etiquetagem morfossintática pode ser selecionado um dos três etiquetadores disponíveis: Freeling, TreeTagger e Padrão. Se a escolha for por análise léxico semântica, pode ser selecionada uma das três WNs disponíveis: PULO, ONTO.PT e OpenWordNet-PT.

Após escolhido o serviço, pode-se executá-lo e aguardar a finalização do processo. Após finalizado o processo a caixa “Texto resultante” apresenta o resultado da consulta. Clicando em cada token do texto resultante suas propriedades são exibidas. A Figura 21 apresenta o resultado de uma consulta para o serviço de etiquetagem morfossintática.

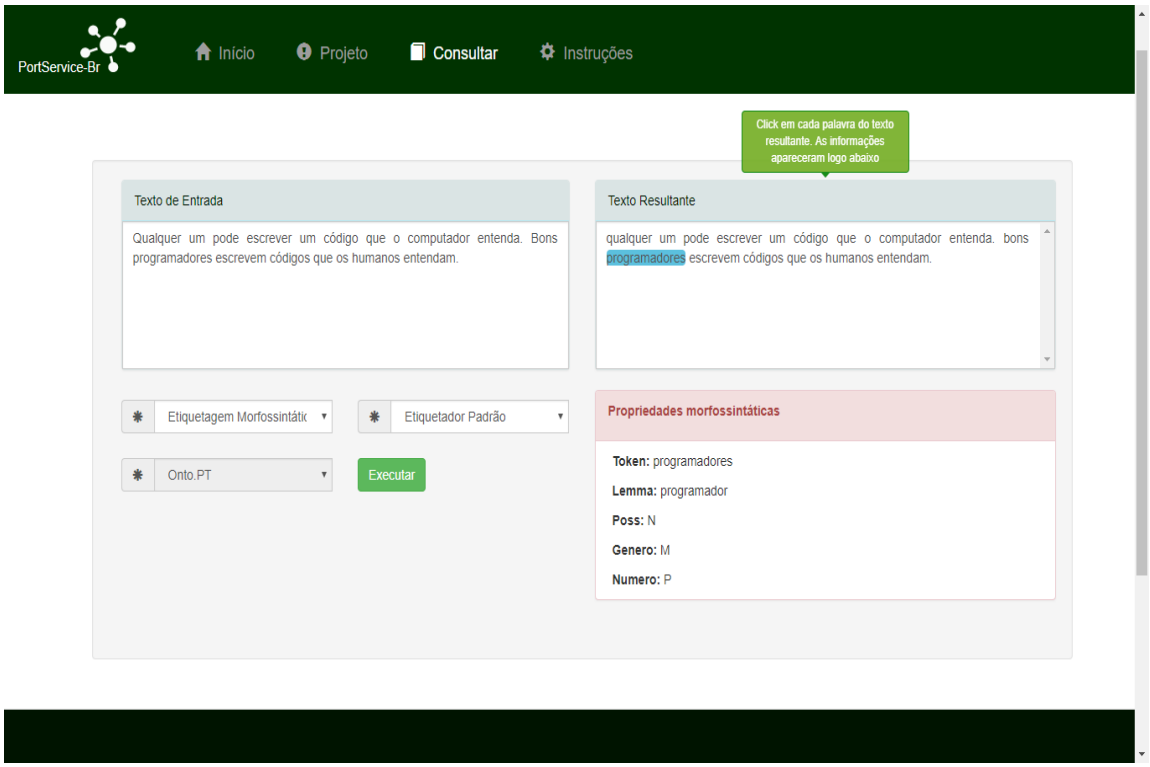

Figura 21. Resultado de uma consulta para o serviço de etiquetagem morfossintática.

Além da interface de consulta, também é possível consultar a interface web de instruções de uso da API REST do PortService-Br. Nessa página é possível obter informações detalhadas sobre o uso de cada um dos serviços oferecidos. A Figura 22 apresenta a página de instruções aos usuários da plataforma PortService-BR, onde é possível obter todas as informações necessárias para acessar as APIs. Podem ser obtidas informações de acesso aos três tipos de serviços atualmente oferecidos na API do PortService-BR.

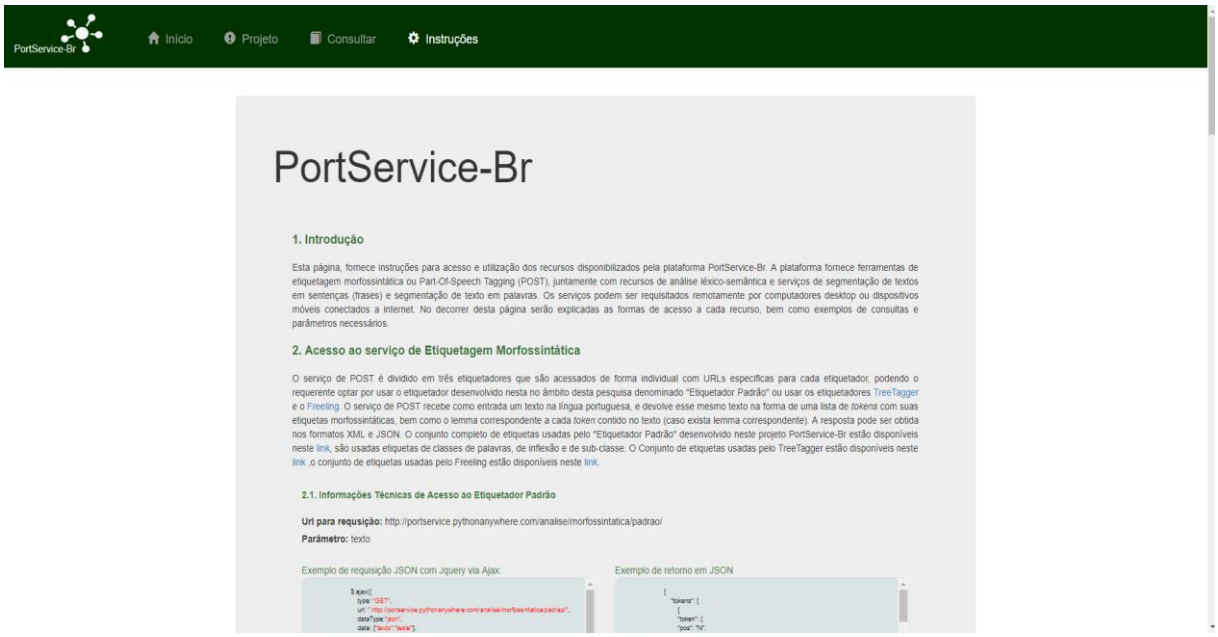

Figura 22. Interface Web de instruções de uso da AP do PortService-Br.

Os próximos capítulos apresentam os resultados alcançados e algumas aplicações que usam os serviços oferecidos na plataforma PortService-Br.

6. RESULTADOS E APLICAÇÕES

Este capítulo é dedicado às aplicações dos serviços desenvolvidos no PortService-Br. Para este fim os serviços da plataforma foram usados em três aplicações distintas: uma API de tradução de Português para LIBRAS, desenvolvida no âmbito desta dissertação; uma aplicação de serviço da PortService-Br no projeto de dissertação de mestrado de Patrícia Teodoro Gaudio Rios, intitulada “Identificando as Implicações Significantes com Suporte Computacional”, e por último o uso de serviços da PortService-Br na plataforma CAP-APL - Plataforma para criação e uso de arquiteturas pedagógicas para aprendizagem de Português e Libras.

6.1 Uso dos serviços da plataforma em uma API de tradução automática Português-Libras

Para testar alguns dos serviços disponibilizados na plataforma PortService-Br, foi desenvolvida uma API de tradução automática da língua Portuguesa para LIBRAS. O mecanismo de memória de tradução foi baseado no sistema de tradução SOTAC desenvolvido no projeto de dissertação de Wesley Lucas Breda. O SOTAC é um sistema de autoria e uso de tradutores automatizados para apoio à tradução, tendo como estudo de caso a tradução de Português para Libras (Breda, 2008). O desenvolvimento da API de tradução foi usado ASP.NET Core 2.0 Microsoft (2017), .NET Standard 2.0 Microsoft (2016), em conjunto com a linguagem de programação C#. O Tradutor desenvolvido usa uma Memória de Tradução (MT), regras baseadas em synsets e regras morfossintáticas, para dinamizar e aumentar a eficiência nas traduções de textos da língua portuguesa para a LIBRAS.

O acesso ao recurso está disponível via HTTP GET, bastando apenas fazer uma requisição para a URL do serviço. A requisição ao serviço de tradução possui o seguinte parâmetro:

- **texto**: Especifica o texto a ser traduzido.

A API se encontra disponível e aberta a consultas no endereço http://apitraducao.azurewebsites.net/api/traducao/portugues/libras/.

6.1.1 Mecanismos de memória de tradução

Ferramentas baseadas em mecanismos de memória de tradução, em grande maioria, ou ainda ferramentas de auxílio do tradutor, usam qualquer par de idiomas. A ideia principal é manter pares de frases e expressões nas línguas trabalhadas pelo tradutor, em um arquivo ou base de dados chamado de memória de tradução. Sempre que o tradutor se deparar com uma frase, palavra ou padrão que esteja na memória de tradução, o mecanismo de memória de tradução nos mostra a tradução desejada. As memórias de tradução também podem apenas servir como referência para um tradutor humano (Somers, 2003).

Os mecanismos de memória de tradução dividem automaticamente os arquivos em segmentos ou partes. Os segmentos são padronizados e definidos a critério do tradutor e geralmente, os segmentos correspondem a frases ou títulos (Somers, 2003). À medida em que a tradução avança, o tradutor interage com a memória de tradução de duas formas.

A primeira é a alimentação, onde cada segmento na língua de origem, junto com o correspondente segmento na língua de destino constitui o que se denomina uma unidade de tradução UT. Essas UTS são armazenadas na memória cada vez que o tradutor passa para o próximo segmento. Ou seja, o tradutor vai gerando a memória ao avançar na tradução.

A segunda fase é a interação e recuperação. Quando uma memória de tradução é aberta em um segmento durante o processo de tradução, o mecanismo busca na memória um segmento igual ou mesmo similar que tenha sido traduzido e armazenado anteriormente. No caso de um resultado positivo da busca, a correspondência entre o segmento que está sendo traduzido e o segmento que foi armazenado na memória pode ser total ou parcial. Havendo uma correspondência total, é apresentada a tradução armazenada e é sugerido sua utilização exatamente na forma em que está armazenada. Então, podemos dizer que existe 100% de correspondência (ou match) entre o segmento do arquivo que se está traduzindo e aquele guardado na memória. Isso implica duas unidades de tradução exatamente iguais.

Também existem casos em que o segmento do arquivo que está sendo traduzido e aquele armazenado na memória seja parcial. Para esses casos, o mecanismo avalia essa correspondência em termos percentuais. É sugerida o uso da tradução armazenada quando o percentual de correspondência for maior ou igual a 75%. Ao sugerir o uso da unidade de tradução similar armazenada, a ferramenta indica qual e onde está a diferença para que o tradutor possa aproveitar a frase incorporando as alterações necessárias (Somers, 2003).

6.1.2 Modelagem da memória de tradução

A modelagem da MT é uma adaptação do modelo proposto por Breda (2008). Algumas adaptações do modelo de Breda foram necessárias, tais como: 1. uso do etiquetador automático usando recurso de etiquetagem morfossintática da plataforma PortService-Br, que realiza essa análise com precisão, levando em consideração possíveis ambiguidades inerentes à língua. No modelo de Breda não foi usado um etiquetador automático; 2. uso de recursos de análise léxico semântica do PortService-Br para identificar expressões sinônimas. Não há esse recurso no modelo de Breda. A Figura 23 apresenta o diagrama de entidade e relacionamento da memória de tradução usada na API.

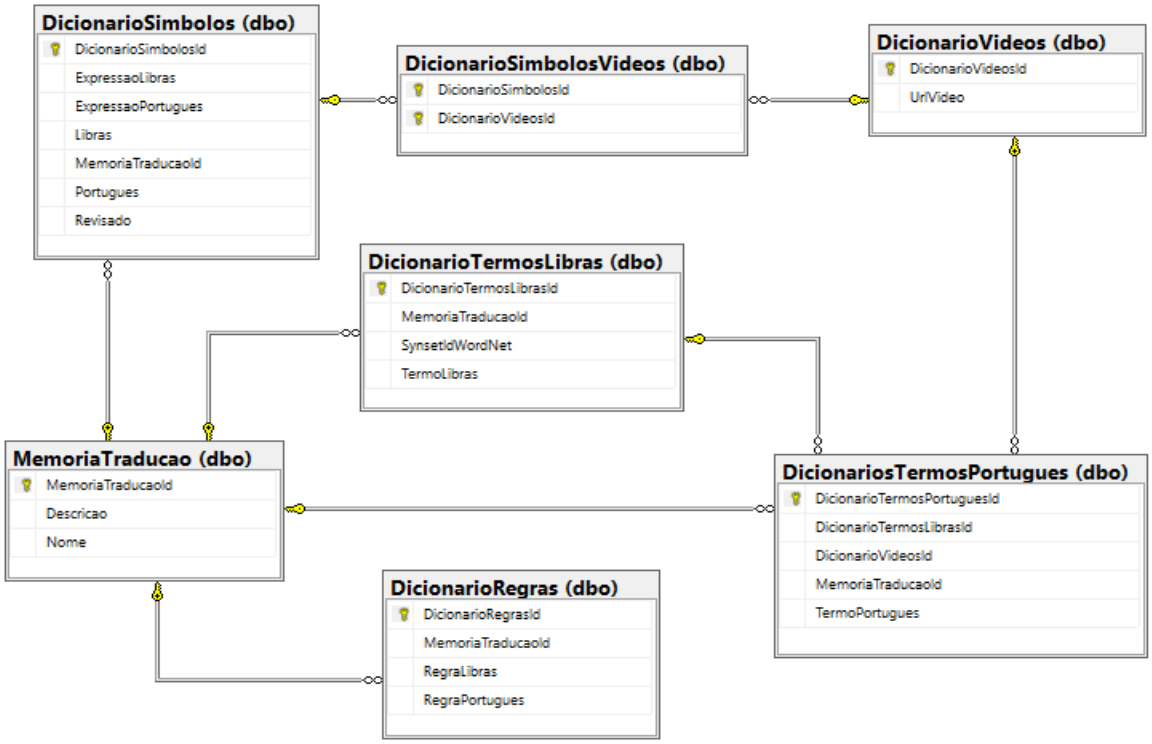

Figura 23. Diagrama de entidade e relacionamento da memória de tradução.

Tabela Memória de tradução: A memória de tradução é uma estrutura onde são armazenadas informações sobre traduções de segmentos de textos ou sobre padrões de tradução entre duas línguas diferentes. Podem existir muitas MTs com diferentes parâmetros e regras para a tradução. Essa tabela é preenchida de forma manual pelos administradores do sistema.

Dicionário de símbolos: O dicionário de símbolos armazena as sentenças na língua portuguesa e sua glosa LIBRAS correspondente. Nesse dicionário também são armazenadas as expressões em forma de synsets da WN e tags sintáticas de cada palavra contida na sentença original e também na glosa LIBRAS. Esse dicionário é o mais importante de toda a MT.

Por meio dele é possível encontrar traduções equivalentes comparando os synsets da expressão. Ele é populado de forma automática pelo sistema.

Dicionário de regras: No dicionário de regras são armazenadas as regras sintáticas de tradução entre a língua portuguesa e a LIBRAS. Essas regras são usadas para traduzir sentenças sem correspondência no Dicionário de Símbolos. Quando não é possível inferir uma tradução por similaridade de expressão é usada a tradução baseada em regras sintáticas. Essa tabela é populada de forma manual pelos administradores do sistema, geralmente linguistas especializados em tradução entre Português e LIBRAS, e pelo próprio sistema.

Dicionário de termos em Português: nesse dicionário são armazenados os terminais em português que podem ser traduzidos para LIBRAS. Essa tabela é preenchida de forma manual pelos administradores do sistema, geralmente linguistas especializados em tradução entre Português e LIBRAS e também pelo próprio sistema.

Dicionário de termos LIBRAS: No dicionário de são armazenados os terminais em LIBRAS e seus respectivos *synsets* da WN. Um terminal LIBRAS pode estar associado a muitos terminais em português. Essa tabela é populada de forma manual pelos administradores do sistema, geralmente linguistas especializados em tradução entre Português e LIBRAS.

Dicionário de vídeos: No dicionário de vídeos são armazenados os vídeos LIBRAS correspondentes a cada terminal do Dicionário de Termos. Dessa forma é garantido que não haja terminal sem um ou mais vídeos de gesticulação LIBRAS associado. Esse dicionário é construído de forma manual pelos administradores do sistema.

Dicionário de símbolos e vídeos: Neste dicionário são armazenadas as relações de sequência de vídeos de cada expressão armazenada no Dicionário de Símbolos, ou seja, para cada sentença, uma sequência de vídeos. Esse dicionário é criado e atualizado de forma automática pelo sistema.

6.1.3 Módulos de tradução do sistema

São usadas até o momento cinco módulos para a tradução automática. Eles implementam os seguintes procedimentos de tradução: 1) a mais simples procura por traduções anteriores comparando uma sentença inteira, 2) comparações de similaridade de expressões baseadas em *synsets* da WN e *tags* sintáticas, 3) tradução baseada em regras sintáticas, 4) tradução palavra por palavra e 5) tradução por datilologia.

Uma tarefa comum a todos os módulos de tradução é a divisão do texto de entrada em sentenças menores. A tradução é realizada à nível de sentença. Essa tarefa é realizada por meio de requisição remota ao serviço de segmentação de sentença da plataforma PortService-Br. Essa funcionalidade utiliza métodos da biblioteca *Natural Language Toolkit* (NLTK) que usa parâmetros de divisão de sentenças da língua portuguesa. A plataforma devolve como retorno uma lista de sentenças no formato json, a serem traduzidas com as diferentes estratégias.

6.1.3.1 Módulo de tradução por igualdade de sentença

Neste módulo de tradução, cada sentença da lista de sentença, é dada como entrada para pesquisa direta. Consiste em pesquisar uma sentença exatamente igual a sentença de entrada. Porém somente as sentenças da MT que foram validadas por linguistas podem fazer parte da pesquisa direta.

Dessa forma foi criada uma *view*[5] do Dicionário de Símbolos, que é composta de

[5] Em teoria de banco de dados, uma visão, ou vista (em inglês: view), conjunto resultado de uma consulta

todas as sentenças cujo campo “Revisado” for verdadeiro, dessa forma torna a pesquisa mais rápida, segura e eficaz, pois somente traduções válidas são retornadas. Essa estratégia parte do princípio de que se existe uma tradução para a sentença, os outros módulos mais sofisticados não precisam ser usados.

Quando encontrada uma sentença idêntica na memória de tradução, significa que existe uma tradução para a sentença de entrada, devolvendo assim a glosa LIBRAS equivalente e a lista de vídeos na sequência correta. Ambos são adicionados à lista de tradução, que armazena a tradução de todas as sentenças da lista de sentenças.

6.1.3.2 Módulo de tradução por similaridade de expressão

Esse módulo é acionado quando o Módulo de Tradução por Igualdade de Sentença não encontra uma sentença exatamente igual a sentença de entrada. Primeiramente são obtidos os *synsets* e *tags* sintáticas de cada palavra da sentença de entrada, por meio de requisição ao serviço de análise léxico-semântica da plataforma PortService-Br. Para essa tarefa a plataforma utiliza a WN OpenWordNet-PT e o etiquetador TreeTagger. A plataforma retorna uma lista de *tokens* no formato json com o *synset* da WN e a etiqueta sintática corretos, para cada palavra da sentença de entrada.

Com a sentença etiquetada é possível montar a expressão que a caracteriza, essa expressão é dada como entrada para pesquisa de expressões similares. Essa estratégia parte do princípio de que, se existe uma expressão equivalente, existe uma tradução equivalente. A Tabela 9 apresenta um exemplo de equivalência de expressões.

Tabela 9. Exemplo de equivalência de Expressões.

Sentença de Entrada	Expressão de Saída
A mulher vê o lobo	< DA0;10243137;2167052;DA0;2114100>
A dama vê o lobo	< DA0;10243137;2167052;DA0;2114100>

armazenada sobre os dados, em que os usuários do banco de dados podem consultar simplesmente como eles fariam em um objeto de coleção de banco de dados persistente.

Na Tabela 9 pode ser observado que as expressões que identificam cada sentença são as mesmas, ou seja, o conjunto de *synsets* e *tags* sintáticas são equivalentes e ambas podem ser traduzidas da mesma forma, com a mesma gesticulação em LIBRAS. Quando é encontrada uma expressão equivalente, o sistema retorna a glosa LIBRAS e a lista de vídeos na sequência correta, ambos são adicionados à lista de tradução, que armazena a tradução de todas as sentenças.

O módulo de tradução por similaridade de expressão, ao encontrar uma tradução satisfatória, faz a partir das tags sintáticas de cada *token*, inferência de regras (usando os padrões com os tags e synsets), montando a sequência de tags e armazena as mesmas no dicionário de regras do sistema, baseando-se na posição de cada *token* da tradução.

6.1.3.3 Módulo de tradução baseada em regras sintáticas

O módulo de tradução baseada em regras é acionado quando o Módulo de Tradução por Similaridade de Expressão não encontra uma expressão equivalente na memória de tradução. Com a lista de *tokens* já etiquetados anteriormente, é possível montar a expressão sintática para a sentença de entrada. A expressão sintática é dada como entrada para uma pesquisa no Dicionário de Regras. Se existir uma regra para a expressão sintática de entrada, significa que é possível traduzir a sentença a partir de regras. A Tabela 10 apresenta uma regra de tradução simples, porém eficiente.

Tabela 10. Regras de Tradução.

Entrada da regra (Português)	Saída da regra (LIBRAS)
DA0;NCFS;VMI;DA0;NCFS	NCFS;VMI;NCFS

Tendo como entrada para a tradução a sentença “A mulher anda na chuva” é produzida a mesma regra de entrada da Tabela 10, sendo assim a saída para essa tradução seria a glosa LIBRAS “MULHER ANDAR CHUVA”. Cada palavra da glosa é dada como entrada para uma pesquisa no Dicionário de Termos LIBRAS, onde cada palavra é diretamente relacionada ao respectivo vídeo do Dicionário de Vídeos, de

modo a recuperar o vídeo correspondente. Ao final é retornada a glosa LIBRAS e a lista de vídeos na sequência correta. A tradução por transferência sintática proposta considera casos em que frases contém várias ocorrências de uma mesma estrutura sintática.

Após a tradução baseada em regras ser executada e bem-sucedida, é preciso armazenar as informações de tradução no Dicionário de Símbolos, para facilitar traduções posteriores. Primeiramente são montadas as expressões com *synsets* da WN e as etiquetas sintáticas, a partir da sentença de entrada e da glosa LIBRAS geradas. Então são armazenados na base de dados ambas as expressões, a sentença de entrada e a glosa LIBRAS, porém o campo “Revisado” da tabela é marcado como falso. Esse novo registro ainda precisa passar pela revisão de um linguista especializado em tradução de Português para LIBRAS, antes de ser liberado para uso na MT.

6.1.3.4 Módulo de tradução palavra por palavra

O módulo de tradução palavra por palavra é executado quando o módulo de tradução baseado em regras não encontra uma regra satisfatório na memória de tradução. Este módulo não leva em conta a semântica das palavras, é uma tradução um pouco mais primitiva.

O módulo recebe como entrada a lista de *tokens* etiquetados nos módulos anteriores. Primeiramente é realizada uma pesquisa a nível de *synset*, na tabela de termos em libras, dessa forma é possível recuperar a glosa e o vídeo associados ao *synset*, que são armazenados na lista de tradução.

Caso a palavra associada a *synset* da lista de tokens não exista no dicionário de termos em português, ela é adicionada ao dicionário, dessa forma o próprio sistema pode alimentar sua base de dados. Ao final são retornados os vídeos e glosas de cada palavra contida na sentença.

6.1.3.5 Módulo de tradução por datilologia

O módulo de tradução por datilologia é executado quando o módulo de tradução palavra por palavra não encontra uma tradução satisfatória. O módulo recebe como entrada a lista de *tokens* etiquetados nos módulos anteriores.

Cada palavra é convertida em uma lista de caracteres, cada caractere é dado como entrada para pesquisa no dicionário de termos em português, onde são recuperados os vídeos e a glosa que os representa, podendo ser números ou letras, que são inseridos na lista de tradução.

A datilologia não é a forma ideal de tradução para LIBRAS, esse módulo somente é usado em casos extremos, quando o sistema não é capaz de traduzir com os outros módulos. A Figura 24 apresenta o diagrama de fluxo do processo de tradução.

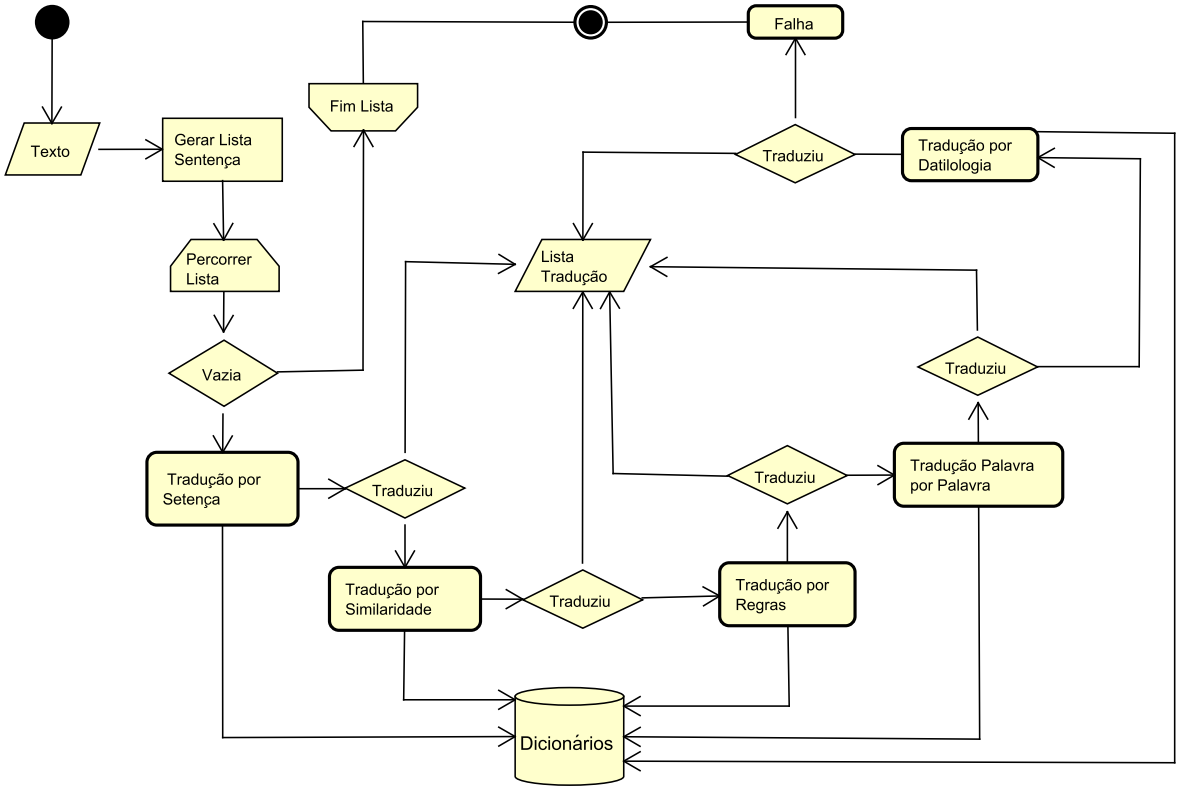

Figura 24. Diagrama de fluxo do processo de tradução.

As sentenças que não puderem ser traduzidas por nenhum dos três procedimentos descritos, são armazenadas no dicionário de símbolos para serem apresentadas aos linguistas, em busca da tradução delas e o respectivo acréscimo no Dicionário de Símbolos e no Dicionário de Regras. Eventualmente, o Dicionário de Vídeos também precisa ser ampliado.

Inicialmente a MT conta somente com regras sintáticas produzidas por linguistas, e a

partir da utilização do sistema, a tabela Dicionário de Símbolos vai sendo populada e revisada constantemente. Este fato permite uma maior confiabilidade nas traduções, à medida que o sistema seja usado.

6.1.4 Testes dos módulos de tradução proposta

Para o teste dos módulos de tradução proposta, foi desenvolvida uma aplicação mobile multiplataforma (Android, IOS), usando a Xamarin.Forms - uma plataforma originalmente criada como kit de ferramentas UI, que permite aos desenvolvedores criar facilmente interfaces de usuário passíveis de serem compartilhadas entre Android, IOS e Windows Phone. As interfaces de usuário são processadas usando os controles nativos da plataforma de destino, permitindo que os aplicativos Xamarin.Forms tenham a aparência e o estilo adequados para cada plataforma (Petzold, 2016).

O aplicativo recebe como entrada um texto digitado pelo usuário, e por meio de requisição à API de tradução passando como parâmetro o texto digitado pelo usuário, recebe como retorno uma glosa com e uma sequência de vídeos que são exibidos na tela do usuário.

Foi selecionada e dada como entrada para o tradutor uma frase retirada do Dicionário da Língua Brasileira de Sinais disponibilizado pelo INES[6], a saber: “Um homem rico tem seu próprio avião”. Essa frase de exemplo produz a seguinte glosa LIBRAS: “HOMEM RICO TER PRÓPRIO AVIÃO”.

Como citado anteriormente a MT, em sua fase inicial, conta somente com regras sintáticas do Dicionário de Regras. A regra sintática que caracteriza a sentença de entrada, levando em consideração as *tags* sintáticas que são produzidas pelo etiquetador TreeTagger e retornados por consulta remota à plataforma PortService-Br, é expressa por: DI0;NCMS;AQ0;VMI;DP3;AQ0;NCMS. A tradução para LIBRAS desta sentença é expressa por: NCMS;AQ0;VMI;AQ0;NCMS.

Inicialmente é realizada a tradução da sentença de entrada com uso do módulo de

---

[6] INES - Instituto Nacional de Educação de Surdos - RJ
https://www.pciconcursos.com.br/concurso/ines-instituto-nacional

tradução baseada em regras, pois a Tabela Dicionário de Símbolos se encontra vazia, que produz a mesma glosa e sequência de vídeos apresentados no dicionário INES. Esse fato revela que o sistema já em sua fase inicial produz traduções satisfatórias. A Figura 25 apresenta a aplicação Android e a aplicação IOS exibindo a sequência de vídeos da tradução.

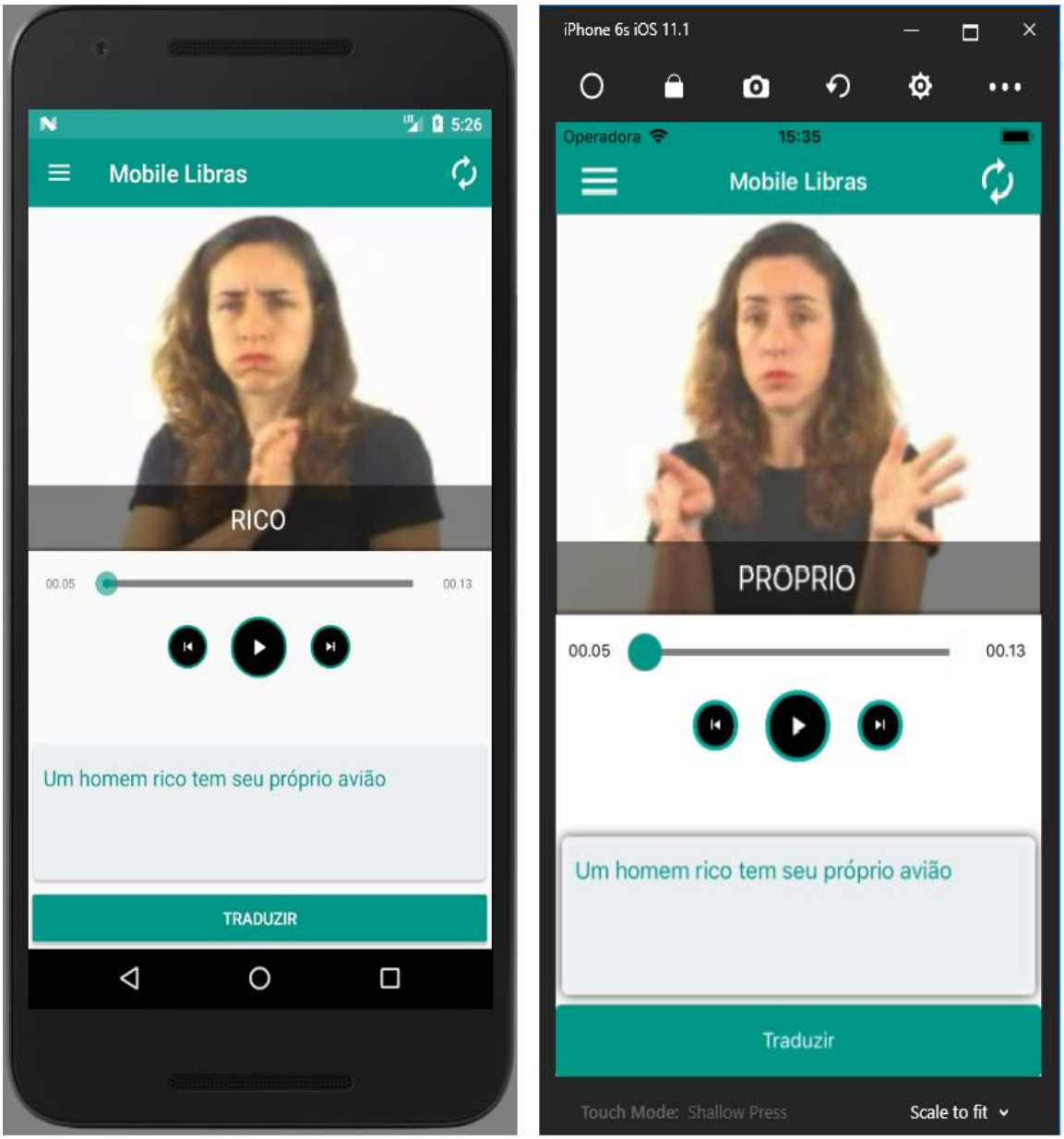

Figura 25. Aplicação Android e a aplicação IOS exibindo a sequência de vídeos da tradução.

Após a execução da tradução baseada em regras o sistema infere as expressões baseada em *synsets* da WN e *tags* sintáticas, que caracterizam tanto a sentença de entrada quanto a glosa LIBRAS geradas, que junto a outras informações sobre a

sentença de entrada são armazenadas no Dicionário de Símbolos da MT. A Tabela 11 apresenta as duas expressões resultantes.

Tabela 11. Expressões resultantes da sentença de teste.

Expressão Sentença de Entrada	Expressão Glosa LIBRAS
DI0;9908025;2024143;2700867;DP3;1104889;2686568	9908025;2024143;2700867;1104889;2686568

Para validar a técnica de tradução por similaridade de expressão, foram dadas como entrada para o tradutor três frases similares a frase inicial a saber:

1) Um homem rico possui seu próprio aeroplano.

2) Um homem abastado possui seu próprio aeroplano.

3) Um moço abastado tem seu próprio aeroplano.

Primeiramente o sistema verifica se existe uma sentença idêntica a sentença de entrada no Dicionário de Símbolos, ou seja, é executado o Módulo de Tradução por Igualdade de Sentença. Logicamente até o momento não existe sentenças iguais as três sentenças acima. Dessa forma a tarefa de tradução é repassada para o Módulo de Tradução por Similaridade de Expressão. Esse módulo de imediato faz uma requisição remota ao serviço de análise léxico-semântica da plataforma PortService-Br, passando como parâmetros as sentenças de entrada.

O retorno da requisição é uma lista de *tokens* com seus respectivos *synset* da WN e *tags* sintáticas, que são usados para montar a expressão correspondente a cada sentença. A Tabela 12 exibe a expressão gerada pelo sistema para cada sentença.

Tabela 12. Expressões resultantes geradas pelo sistema.

Identificador	Expressão Resultante
1	DI0;9908025;2024143;2700867;DP3;1104889;2686568

2	DI0;9908025;2024143;2700867;DP3;1104889;2686568
3	DI0;9908025;2024143;2700867;DP3;1104889;2686568

Como pode ser visualizado na Tabela 12, todas as sentenças tiveram como resultado a mesma expressão, que é idêntica a expressão da sentença selecionada para o teste de tradução. Sendo assim, o sistema infere que existe uma tradução equivalente na MT para as expressões de entrada, então é retornada a glosa LIBRAS e a sequência de vídeos na ordem correta para cada sentença.

Apesar de ser uma implementação de teste para a API da plataforma PortService-Br, sem a participação de especialistas em LIBRAS, o tradutor revela um grande potencial, necessitando apenas de um maior aprimoramento das traduções para trabalhos futuros.

6.2 Uso dos serviços da plataforma no projeto CAP-APL

A CAP-APL é uma plataforma web que disponibiliza recursos digitais para a criação e uso de arquiteturas pedagógicas (APs) para a aprendizagem de Português e de Libras, (Tavares, Reinoso e Almeida, 2017).

As APs são suportes estruturantes à aprendizagem, propostos por Carvalho et al. (2005), com foco na construção do conhecimento, baseadas nas ideias construtivistas, de Jean Piaget (1985), e no incentivo à autonomia, de Paulo Freire (1988). As APs formam estruturas de aprendizagem compostas de um conjunto base: abordagem pedagógica, software, internet, inteligência artificial, educação à distância e concepção de tempo e espaço. Usam didáticas flexíveis, maleáveis e totalmente adaptáveis a diferentes temas. As perspectivas de tempo e espaço de aprendizagem passam a se moldar aos ritmos impostos pelo sujeito que aprende, removendo a exigência pela localização em escola ou sala de aula.

O Framework Construtor de Arquiteturas Pedagógicas (CAP) possibilita a criação de arquiteturas pedagógicas virtuais, permitindo a um professor, mesmo leigo em informática, gerar um conjunto de atividades e implementar uma AP com recursos digitais que a suportem. Esse Framework permite ao professor alocar recursos e

configurá-los, do modo mais adequado, para suportar a AP especificada por ele (Reinoso et al, 2016). O CAP-APL foi desenvolvido para suprir as demandas de Processamento de Linguagem Natural, possibilitando trabalhar a aprendizagem de Português e LIBRAS, importantes para formação de intérpretes e tradutores LIBRAS.

6.2.1 Recursos digitais para a aprendizagem de Português e Libras

Os recursos digitais usados na aprendizagem de Libras estão apresentados em Reinoso et al (2016). Os recursos digitais usados na aprendizagem de Português são selecionados da plataforma PortService-Br. Esses serviços são apresentados a seguir:

1. Etiquetador de palavras, que recebe um texto em português e retorna o texto com suas palavras etiquetadas (classe gramatical, gênero, número, entre outros);

2. Lematizador que recebe uma palavra e retorna seu lema correspondente;

3. Gerador de synset (um identificador de uma classe de palavras sinônimas) que recebe uma palavra e, com a ajuda de uma WordNet (WordNet, 2015), retorna o synset dessa palavra;

4. Apresentador da classe de um synset: recebe um synset e retorna as palavras sinônimas representadas por ele.

A plataforma CAP-APL também fez uso da API de tradução de Português para Libras, apresentada nas seções anteriores. Usando a plataforma CAP que implementa um modelo de construção de APs, equipada com os recursos digitais adicionais, como um gravador de vídeos, um visualizador de vídeos, um visualizador de textos, um editor de textos e uma planilha, foi criada a CAP-APL, uma plataforma para a criação e uso de arquiteturas pedagógicas para a aprendizagem de Português e Libras.

6.2.2 Arquiteturas pedagógicas criadas com uso dos serviços do PortService-Br

Para validação dos recursos oferecidos do CAP-APL foram instanciadas duas APs,

uma para a aprendizagem de Português que usa o serviço de etiquetagem morfossintática do PortService-Br e uma para a aprendizagem de tradução de Português para Libras que usa a API de tradução de Português-LIBRAS mencionada na seção anterior, ou seja, direta ou indiretamente, todos os serviços da plataforma PortService-Br estão sendo usados no CAP-APL.

6.2.2.1 AP1 - Arquitetura Pedagógica "Aprendendo a etiquetar palavras de textos em português"

Segue a descrição da estrutura da AP1:

Objetivo de aprendizagem: Aprender a classificar as palavras da língua portuguesa segundo sua distribuição morfossintática (classe gramatical, gênero, número).

Atividade: Analisar textos da língua portuguesa e etiquetar cada palavra segundo sua distribuição morfossintática.

Método: A partir de um texto disponível na AP, os alunos devem fazer a etiquetagem morfossintática das palavras dele. Os aprendizes podem usar o etiquetador morfossintático TreeTagger (recurso digital disponível), de modo a analisarem a etiquetagem das palavras do texto e anotarem dúvidas a serem discutidas com o professor.

Recursos digitais: Um visualizador e um etiquetador (etiquetador morfossintático TreeTagger, da biblioteca de recursos digitais da CAP-APL) de texto em Português; um visualizador de texto para a apresentação do texto etiquetado.

Exemplo de uso da AP1: Para exemplificar o uso do etiquetador de palavras, foi dada como entrada a sentença "A mulher vê o lobo", que produz a seguinte sequência de tags: "<DA0;NCFS;VMI;DA0;NCMS>". A Tabela 13 apresenta detalhadamente o significado de cada uma dessas tags.

Tabela 13. Significado das tags geradas pelo etiquetador.

Tag	Significado
DA0	Determinante (Artigo)
NCFS	Substantivo comum feminino singular

VMI	Verbo principal no modo indicativo
DA0	Determinante (Artigo)
NCMS	Substantivo comum masculino singular

6.2.2.2 AP2 - Arquitetura Pedagógica “Traduzindo expressões do Português para Libras”

Segue a descrição da estrutura da AP2:

Objetivo de aprendizagem: Aprender a traduzir expressões do Português para Libras.

Atividade: Traduzir expressões do Português para Libras.

Método: O estudante grava um vídeo com a gesticulação, feita por ele, da expressão Libras correspondente à expressão em Português fornecida. O estudante pode usar o tradutor de Português para Libras para visualizar a tradução das expressões em Português para Libras, de modo a aprender a tradução correspondente.

Recursos digitais: Um gravador de vídeos e um tradutor de Português para Libras, da Biblioteca de Recursos Digitais da CAP-APL (neste caso, a API de tradução Português-Libras da PortService-BR).

Exemplo de uso da AP2: Na tradução de Português para Libras da sentença “A mulher vê o lobo”, o tradutor usa o padrão definido pela sequência de tags “<DA0;10243137;2167052;DA0;2114100>”. Esse padrão descreve sentenças equivalentes à sentença de entrada. A Tabela 14 apresenta detalhadamente o significado de cada tag da expressão de saída.

A sequência de tags acima é então convertida para uma expressão de equivalência em LIBRAS, onde são eliminados os determinantes, gerando a sequência de tags <10243137;2167052;2114100>, equivalente à glosa “MULHER VER LOBO” que é exibida via uma sequência de vídeos, com a forma gestual animada em Libras dessa glosa. Uma frase como “a mulher observa o lobo” também geraria a mesma tradução, pois são parte do mesmo conjunto de synsets sinônimos.

Tabela 14. Significado das tags usadas no tradutor.

Tag	Significado
DA0	Determinante (Artigo)
10243137	Synset da WordNet referente ao conjunto de substantivos sinônimos que incluem o substantivo” mulher”.
2167052	Synset da WordNet referente ao conjunto de verbos sinônimos que incluem o verbo “ver”.
DA0	Determinante (Artigo)
2114100	Synset da WordNet referente ao conjunto de substantivos sinônimos que incluem o substantivo “lobo”.

Ambas as arquiteturas foram testadas e avaliadas em salas de aulas reais com boa aceitação tanto por parte dos alunos, quanto dos professores, com pontuação média de 70% em termos de usabilidade do sistema e facilidade de uso das ferramentas oferecidas. Os serviços oferecidos tanto pela API de tradução de Português para Libras, quanto dos serviços oferecidos pelo PortService-Br tiveram boa aceitação.

6.3 Uso dos serviços da plataforma na Identificação de Implicações Significantes com Suporte Computacional

O objetivo do projeto foi desenvolver uma ferramenta de suporte computacional denominada CogniS - um acrônimo de Cognição Significante, para a identificação das implicações significantes de Piaget em mapas conceituais, dotados da capacidade de demonstrar os níveis de conhecimento de um dado domínio (Rios, 2016).

Os mapas conceituais são representações gráficas de relações entre conceitos. São usados em diferentes domínios do conhecimento. Em particular, eles têm atraído o interesse dos educadores em todo o mundo. Novak e Cañas (2006), definem mapas conceituais como ferramentas para organizar e representar o conhecimento. Constituem uma rede de nós, que representam conceitos ou objetos, que são conectados por arcos com rótulos descritores das relações entre os nós.

Para Piaget, “um sistema conceitual, com efeito, é um sistema tal que seus elementos se apoiam inevitavelmente uns nos outros, sendo ao mesmo tempo aberto a todas as trocas com o exterior. As implicações significantes definidas por Piaget se classificam em três maneiras. Ele cita uma evolução significante de três tipos de inferências (PIAGET, 1985).

1. As antecipações limitadas para permitir a repetição de objetos observáveis, ou modificações, já observadas empiricamente. A este nível o sujeito não irá raciocinar e inferir, mas fará referência a um mundo de objetos empíricos;

2. Inferências a respeito de determinadas antecipações que vão além do observável e são fundadas em implicações necessárias, mas ainda não exibindo suas “razões”. São aqui implicações entre as ações que são geradas pela abstração reflexiva e não se limitam a tirar as consequências lógicas de abstrações empíricas;

3. As inferências com base em “razões” ou possíveis demonstrações.

Piaget propôs ainda uma distinção entre as formas ou grau de implicações significativas em três níveis:

a) Implicações Locais: a importância das ações é determinada pelos seus resultados apurados. As implicações permanecem restritas a dados muito limitados e contextos específicos.

b) Implicações Sistêmicas: as implicações deste tipo já são inseridas no sistema de relações, mas entendida apenas por etapas sucessivas entre elementos vizinhos. Nesse nível, elas começam julgamentos sobre o que é possível e o que não é. Mas essas inferências não são suficientes para atingir as relações necessárias. Ainda há confusão entre necessidade e generalidade.

c) Implicações Estruturais: são as implicações relacionadas com as composições internas das estruturas já construídas. Neste nível há entendimento endógeno do “porquê” dos fatos gerais observados. As relações se tornam necessárias.

O projeto CogniS visa a identificação das implicações significantes de Piaget nos mapas conceituais, dotados da capacidade de demonstrar os níveis de conhecimento de um dado domínio. A ideia central consiste em desenvolver uma ferramenta que possibilite a navegação pelas arestas do grafo. A navegação é realizada por uma agente de software que usa as implicações significantes, para a identificação dos vértices e das arestas (Rios, 2016). A Figura 26 apresenta um mapa conceitual gerado pela ferramenta CogniS, com as implicações identificadas. Para as locais, usará o amarelo; para as sistêmicas, o verde; e para as estruturais, o azul.

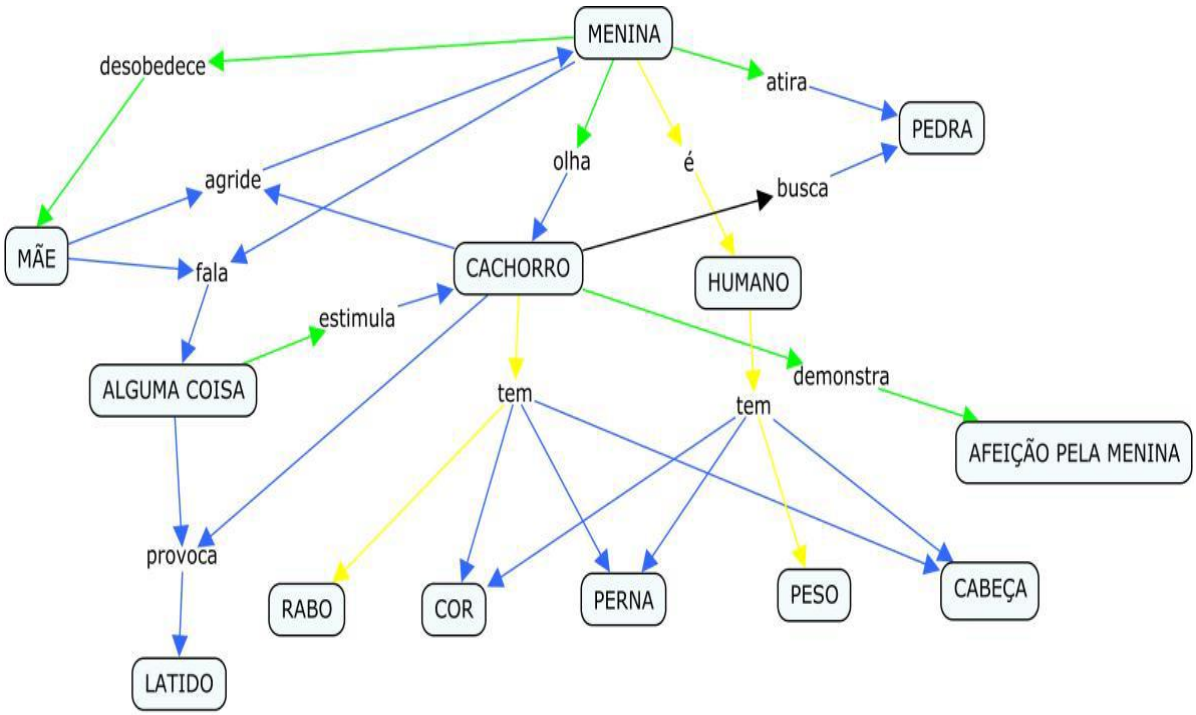

Figura 26. Mapa conceitual gerado pela ferramenta CogniS, com as implicações identificadas.

Para a identificação das implicações locais e sistêmicas é usado apenas o verbo contido na sentença designadora da aresta como padrão de casamento da aresta com a implicação, nesta identificação os verbos precisam primeiramente ser identificados e posteriormente lematizados.

A tarefa de identificação o verbo é realizado por meio de consulta ao serviço de etiquetagem morfossintática da plataforma PortService-Br, que foi responsável por encontrar os verbos contidos na sentença designadora e o lematizar. O uso da

plataforma PortService-Br foi imprescindível para a implementação do protótipo e o sucesso do projeto CogniS.

7 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões e as contribuições científicas alcançadas nesta dissertação. De igual maneira, são apresentados também os trabalhos futuros propostos a partir desta dissertação de mestrado.

7.1 Conclusões

Grandes são os desafios que norteiam o processamento computacional das línguas naturais, sobretudo da língua portuguesa, tanto a nível de desenvolvimento de ferramentas quanto em seu uso. Esta dissertação surgiu, a partir desses desafios, da grande dificuldade detectada no uso, configuração e manutenção de sistemas voltados para o PLN. Muitas das ferramentas existentes são desenvolvidas para sistemas operacionais específicos e para versões específicas deles. Além de possuírem um número muito grande de dependências para seu pleno funcionamento e uma documentação muito técnica e difícil de ser interpretada por usuários não especialista em informática. Também muitas dessas ferramentas são proprietárias, ou seja, seu uso só pode ser realizado mediante pagamento.

Para contornar esses problemas, foi desenvolvida uma plataforma de uso gratuito para oferecer serviços de PLN na web. Essa plataforma permite ao usuário abstrair toda a parte de instalação e configuração, deixando-as ao encargo dos administradores da plataforma.

Em um primeiro momento foi investigado quais ferramentas e recursos de PLN seriam disponibilizados na primeira versão do sistema. Foi decidido, após pesquisas sobre serviços e recursos de PLN mais usados, que seriam oferecidos, no primeiro momento, serviços de etiquetagem morfossintática (usando os etiquetadores Freeling, TreeTagger e um etiquetador desenvolvido no âmbito deste projeto), segmentação de textos em frases e palavras (usando a biblioteca NLTK) e recursos de análise léxico semântica com uso de WordNets da Língua portuguesa (ONTO.PT, PULO e OpenWordNet-PT).

Em um segundo momento foi investigado qual a linguagem de programação seria mais adequada para se trabalhar com PLN. Após vasta pesquisa foi decidido pelo uso da Linguagem Python, que já oferece recursos nativos para o desenvolvimento

de sistemas baseados em PLN, além de contar com muitas bibliotecas para essa tarefa, como o NLTK e o RDFLib.

7.2 Contribuições tecnológicas

Com o projeto delineado foi implementada uma plataforma web Portservice-BR, usando o Web *framework* Django e a biblioteca Django RestFramework, que fornece uma API REST. A plataforma é capaz de carregar em memória todos os módulos do sistema de forma dinâmica. Para cada tipo de serviço é oferecido uma URL de consulta via HTTP GET, bastando apenas ao usuário passar um parâmetro correspondente ao texto ou palavra a ser submetida à plataforma.

7.3 Contribuições científicas

O PortService-Br consegue abstrair todas as questões de instalação, configuração e manutenção das ferramentas por parte dos usuários.

A plataforma PortService-Br foi testada em três aplicações que fazem uso de PLN, obtendo bons resultados em todos os serviços oferecidos, a saber:

- Uma API de tradução automática de Português para LIBRAS, desenvolvida no âmbito desta dissertação de mestrado, onde foram usados todos os serviços oferecidos na plataforma;

- No âmbito do projeto CAP-APL, uma plataforma web que disponibiliza recursos digitais para a criação e uso de arquiteturas pedagógicas (APs) para a aprendizagem de Português e de Libras, onde foram usados os serviços de etiquetagem morfossintática e análise léxico semântica;

- No âmbito do projeto CogniS, uma ferramenta computacional para a identificação das implicações significantes, segundo Piaget, em mapas conceituais, dotada da capacidade de demonstrar os níveis de conhecimento de um dado domínio, onde foi usado o serviço de etiquetagem morfossintática.

De acordo com os testes realizados com o PortService-Br, verificou-se que, mesmo em um estágio inicial, produziu um efeito significativo com base em bons resultados reportados pelos sistemas que testaram os serviços oferecidos. O objetivo foi

alcançado e o modelo proposto na plataforma bem como no conjunto de características estabelecidas, gerou um sistema eficaz, competitivo, sendo ainda passível de fácil adaptação, modificação e inserção de novos serviços.

Pode-se citar, dentre as contribuições científicas desta dissertação, além do próprio PortService-Br, também o etiquetador morfossintático desenvolvido denominado etiquetador padrão, que em testes realizados, chegou a uma acurácia de 98% com a técnica empregada. Esta dissertação também contribuiu com o desenvolvimento de uma API de tradução automática de Português para Libras, que também apresenta bons resultados nas traduções usando o PortService.Br.

A seguir faz-se uma retomada das questões de investigação, listadas no capítulo 1, que se buscava responder ao longo deste trabalho.

A QI1 (“Existem plataformas de PLN semelhantes a proposta nesta dissertação?”) teve como resultado a descoberta de três plataformas similares a proposta desta dissertação, porém são plataformas proprietárias de uso comercial, necessitando de prévio cadastro para uso.

A QI2 (“Quais recursos e ferramentas podem ser oferecidos pela plataforma?”): a plataforma PortService-Br foi desenvolvida de modo a oferecer serviços modulares, assim facilitando a adição de novas ferramentas e funcionalidades. Dessa forma a plataforma pode oferecer muitos outros serviços além dos atualmente oferecidos.

A QI3 (“Como usar os serviços oferecidos pela plataforma em aplicações computacionais?”): o desenvolvimento da plataforma como uma API REST facilita seu uso por qualquer tipo de aplicativo com acesso a internet, com esforço mínimo.

A QI4 (“Os serviços oferecidos podem ser usados em aplicações voltadas a informática na educação?”): com uso dos serviços do PortService-Br pela plataforma CAP-APL ficou claro o potencial de uso dos recursos oferecidos para aplicações voltadas para a informática na educação, sobretudo para o aprendizado da língua portuguesa.

A QI5 (“Os serviços oferecidos podem ser usados em tradutores automáticos de Português-Libras?”): o desenvolvimento experimental de uma API de tradução automática Português-Libras, usando os serviços do PortService-Br para processar

os textos, evidenciou sua eficácia nessa tarefa. O tradutor já em seu primeiro estágio de desenvolvimento se mostrou eficaz com as técnicas propostas.

7.4 Trabalhos futuros

Como trabalhos futuros pretende-se aumentar a quantidade de serviços oferecidos, que são bastante importantes e muito usados em aplicações baseadas em PLN, tais como:

Identificação de entidades nomeadas (REN) pode ser definida como uma tarefa cujo objetivo é identificar as entidades nomeadas bem como sua classificação, atribuindo uma categoria semântica e sintática para essas entidades. REN é uma técnica amplamente utilizada no Processamento da Linguagem Natural (PLN) e consiste da identificação de nomes de entidades-chave presentes na forma livre de dados textuais (Amaral, 2013).

Sumarização o objetivo da sumarização é reconhecer, em um texto, o que é relevante e o que pode ser descartado, para compor um sumário. Esse é um ponto problemático e sujeito a análise minuciosa por parte dos pesquisadores (Martins et al., 2001).

Análise de sentimentos ou Mineração de Opinião visa descobrir de forma computacional opiniões e seus conceitos relacionados, como: sentimentos, atitudes, avaliações e emoções relacionadas a alguma entidade, tais como: produtos, serviços, organizações, indivíduos, eventos, tópicos e seus atributos (F Filho et al., 2015).

Também são trabalhos futuros a implementação de mecanismos de processamento de arquivos inteiros, por meio de *uploads* para diversos formatos. No estado atual de desenvolvimento a plataforma somente processa pequenos fragmentos de textos via HTTP GET.

Implementar mecanismos de tratamento de tempos verbais na API de tradução Português-Libras, no estado atual o tradutor ainda não conta com esse tipo de tratamento bastante importante para compreensão das traduções por deficientes auditivos. A Libras não tem em suas formas verbais a marca de tempo como o português. Essas formas podem se modular para aspecto. Algumas delas também

se flexionam para número e pessoa. Dessa forma, quando o verbo se refere há tempo passado, futuro ou presente, o que vai marcar o tempo da ação ou do evento serão itens lexicais ou sinais adverbiais como ontem, amanhã, hoje, semana-passada, semana que vem. Com isso, não há risco de ambiguidade porque se sabe que o que está sendo narrado iniciou-se com uma marca no passado, enquanto não aparecer outro item ou sinal para marcar outro tempo, tudo será interpretado como tendo ocorrido no passado (Crato, 210).

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, Daniela Oliveira Ferreira do**. O reconhecimento de entidades nomeadas por meio de conditional random fields para a língua portuguesa**. 2013. Dissertação de Mestrado. Pontifícia Universidade Católica do Rio Grande do Sul.

BREDA, W. L. **Um Ambiente para Apoio à Tradução Baseado em Conhecimento: Estudo de Caso com Português-Libras**. 2008. Dissertação de Mestrado. Universidade Federal do Espírito Santo.

BOND, Francis; FOSTER, Ryan. **Linking and Extending an Open Multilingual Wordnet.** In: ACL (1). 2013. p. 1352-1362.

Brill, Eric. **A simple rule-bases part of speech tagger**. ANLP, Trento, 1992.

CARVALHO, M. J. S., NEVADO, R. A., e MENEZES, C. S. (2005) **"Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático"**, In: XVI Simpósio Brasileiro de Informática na Educação, Juiz de Fora-MG. Brasil. págs. 351-360.

CASS, Stephen. **The 2015 Top Ten Programming Languages**. Disponível em:<http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages>.Acesso em: novembro 2016.

CHRISTIE, Tom. **API Guide**. Disponível em: <http://www.django-rest-framework.org/>. Acesso em: novembro 2015.

CRATO, Aline Nascimento et al. **Marcação de tempo por surdos sinalizadores brasileiros**. Pró-Fono Revista de Atualização Científica, v. 22, n. 3, p. 163-168, 2010.

DANSO, Samuel; LAMB, William. **Developing an Automatic Part-of-Speech Tagger for Scottish Gaelic**. CLTW 2014, p. 1, 2014.

DE PAIVA, Valeria et al. **NomLex-PT: A Lexicon of Portuguese Nominalizations**. In: LREC. 2014. p. 2851-2858.

DE PAIVA, Valeria; RADEMAKER, Alexandre; DE MELO, Gerard**. OpenWordNet-PT: An Open Brazilian Wordnet for Reasoning**. In: COLING (Demos). 2012. p. 353-360.

DJANGO SOFTWARE FOUNDATION. **Meet Django**. Disponível em:<https://www.djangoproject.com/>. Acesso em: novembro 2017.

DE LIMA, Vera Lúcia Strube; NUNES, Maria das Graças Volpe; VIEIRA, Renata**. Desafios do processamento de línguas naturais**. SEMISH-Seminário Integrado de Software e Hardware, v. 34, p. 1, 2007.

DOMINGUES, Miriam Lúcia; FAVERO, Eloi Luiz; DE MEDEIROS, Ivo Paixão. **O desenvolvimento de um etiquetador morfossintático com alta acurácia para o português**. Avanços da Linguística de Corpus no Brasil. São Paulo: Humanitas, p. 267-286, 2008.

DOMINGUES, Miriam Lúcia Campos Serra. **Abordagem para o desenvolvimento de um etiquetador de alta acurácia para o Português do Brasil**. Tese de doutoramento, Universidade Federal do Pará, 2011.

ERYIGIT, Gülsen. **ITU Turkish NLP Web Service**. In: EACL. 2014. p. 1-4.

F FILHO, Carlos Augusto; DE OLIVEIRA, Fernando M.; DE OM LOBO, Ramon. **Extração de Dados do Twitter para aplicação na Análise de Sentimentos**. Anais do Computer on the Beach, p. 423-425, 2015.

FELLBAUM, Christiane. **WordNet. In: Theory and applications of ontology**: computer applications. Springer Netherlands, 2010. p. 231-243.

FREIRE, PAulo.(1998) **Pedagogia da Autonomia: saberes necessários à prática educativa**. Rio de Janeiro: Paz e Terra.

GABRIEL JUNIOR, Rene Faustino Gabriel. **UTILIZAÇÃO DA WEB SEMANTICA E RDF EM ESTUDOS MÉTRICOS DA INFORMAÇÃO**: aplicação na base Brapci. In: XVII Encontro Nacional de Pesquisa em Ciência da Informação. 2016.

GAMALLO, Pablo; GARCIA, Marcos. **FreeLing e TreeTagger: um estudo comparativo no âmbito do Português**. Technical report, Universidade de Santiago de Compostela, 2013.

JURAFSKY, Dan; MARTIN, James H. **Speech and language processing**. Pearson, 2014.

MARTINS, Camilla Brandel et al. **Introdução à sumarização automática**. Relatório Técnico RT-DC, v. 2, p. 2001, 2001.

MAZIERO, Erick G. et al. **A base de dados lexical e a 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. ACM, 2008. p. 390-392.

MANNING, Christopher D. **Part-of-speech tagging from 97% to 100%: is it time for some linguistics**. In: International Conference on Intelligent Text Processing and Computational Linguistics. Springer Berlin Heidelberg, 2011. p. 171-189.

MICROSOFT (Estados Unidos). **.NET Standard**. 2016. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/standard/net-standard>. Acesso em: 30 dez. 2016.

MICROSOFT (Estados Unidos). **Introduction to ASP.NET Core**. 2017. Disponível em: <https://docs.microsoft.com/en-us/aspnet/core/>. Acesso em: 30 set. 2017.

NEVES, Ricardo Fernando Muacho Fernandes Lima. **Classificação automática de textos baseada em ontologias**. 2010. Tese de Doutorado. Faculdade de Ciências e Tecnologia.

NOVAK, J. D.; CAÑAS A. J. **The Theory Underlying Concept Maps and How to Construct Them**, Technical Report IHMC CmapTools 2006-01, Institute for Human and Machine Cognition. Available in: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMap.

OLIVEIRA, Hugo Gonçalo et al. **As wordnets do português**. Oslo Studies in Language, v. 7, n. 1, 2015.

OLIVEIRA, Hugo Gonçalo et al. **PAPEL: a dictionary-based lexical ontology for Portuguese**. In: International Conference on Computational Processing of the Portuguese Language. Springer Berlin Heidelberg, 2008. p. 31-40.

OLIVEIRA, Hugo Gonçalo; GOMES, Paulo. **ECO and Onto. PT: a flexible approach for creating a Portuguese wordnet automatically**. Language resources and evaluation, v. 48, n. 2, p. 373-393, 2014.

PADRÓ, Luís; STANILOVSKY, Evgeny. **Freeling 3.0: Towards wider multilinguality**. In: LREC2012. 2012.

PERKINS, Jacob. **Python 3 Text Processing with NLTK 3 Cookbook**. Packt Publishing Ltd, 2014.

PETZOLD, Charles. **Creating Mobile Apps with Xamarin.Forms First Edition**/Petzold C. 2016.

Piaget, J. (1985). **O possível e o necessário – evolução dos possíveis na criança**. Porto Alegre, Artes Médicas.

QIU, Xipeng; ZHANG, Qi; HUANG, Xuanjing. **FudanNLP: A Toolkit for Chinese Natural Language Processing**. In: ACL (Conference System Demonstrations). 2013. p. 49-54.

RADEMAKER, Alexandre et al. **OpenWordNet-PT: a project report**. In: Proceedings of the 7th Global WordNet Conference. 2014. p. 383-390.

Reinoso, L. F.; Almeida, R. F.; Tavares, O. L. (2016) **Modelo CAP: Construtor de Arquiteturas Pedagógicas**. In: Congresso Internacional de Informática Educativa, 2016, Chile. Actas do Congresso Internacional de Informática Educativa. Santiago/Chile: Jaime Sánchez, v. 12, p. 89-94.

RIOS, Patricia Teodoro. **Identificando as Implicações Significantes com Suporte Computacional**. 2016. 84 f. Dissertação (Mestrado) - Curso de Mestrado em Informática, Universidade Federal do Espirito Santo, Vitória, 2016.

RONDON, Alexandre C.; DE MEDEIROS CASELI, Helena; RAMISCH, Carlos. **Never-Ending Multiword Expressions** Learning. In: Proceedings of NAACL-HLT. 2015. p. 45-53.

SCHMID, Helmut. **Probabilistic part-ofispeech tagging using decision trees**. In: New methods in language processing. Routledge, 2013. p. 154.

SIMOES, Alberto; GUINOVART, Xavier Gómez. **Bootstrapping a Portuguese wordnet from Galician, Spanish and English wordnets**. In: Advances in Speech and Language Technologies for Iberian Languages. Springer International Publishing, 2014. p. 239-248.

SOMERS, Harold. **Translation memory systems**. Benjamins Translation Library, v. 35, p. 31-48, 2003.

TAVARES, Orivaldo; REINOSO, Luiz; DE ALMEIDA, Wanderson. **CAP-APL: plataforma para criação e uso de arquiteturas pedagógicas para aprendizagem de Português e Libras**. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2017. p. 466.

THIELE, Pablo Frederico Oliveira. **Desambiguação de anotações morfossintáticas feitas por MTMDD**. 2015.

VAN ROSSUM, Guido. **The History of Python: Introduction and Overview**. Disponível em:<http://python-history.blogspot.com.br/2009/01/introduction-and-overview.html>. Acessoem: novembro 2015.

VIEIRA, Nuno; SIMÕES, Alberto; CARVALHO, Nuno Ramos. SplineAPI: **A REST API for NLP Services**. In: International Symposium on Languages, Applications and Technologies. Springer International Publishing, 2015. p. 205-215.