

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO**

Vinícius Silva Nogueira

**APERTIUM WDM - UMA ALTERNATIVA COLABORATIVA PARA
AMPLIAÇÃO DE DICIONÁRIOS EM FERRAMENTAS DE
TRADUÇÃO AUTOMÁTICA BASEADA EM REGRAS**

Timóteo

2019

Vinícius Silva Nogueira

**APERTIUM WDM - UMA ALTERNATIVA COLABORATIVA PARA
AMPLIAÇÃO DE DICIONÁRIOS EM FERRAMENTAS DE
TRADUÇÃO AUTOMÁTICA BASEADA EM REGRAS**

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Aléssio Miranda Júnior

Timóteo

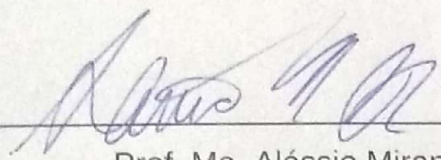
2019

Vinícius Silva Nogueira

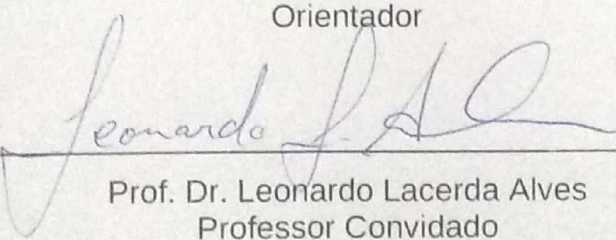
**APERTIUM WDM - UMA ALTERNATIVA COLABORATIVA PARA AMPLIAÇÃO DE
DICIONÁRIOS EM FERRAMENTAS DE TRADUÇÃO AUTOMÁTICA BASEADA EM
REGRAS**

Trabalho de Conclusão de Curso
apresentado ao Curso de Engenharia de
Computação do Centro Federal de Educação
Tecnológica de Minas Gerais, campus Timóteo,
como requisito parcial para obtenção do título de
Engenheiro de Computação.

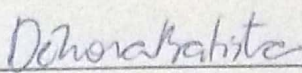
Trabalho aprovado. Timóteo, 11 de dezembro de 2019:



Prof. Me. Aléssio Miranda Júnior
Orientador



Prof. Dr. Leonardo Lacerda Alves
Professor Convidado



Prof. Débora Evelyn Silva Batista
Professor Convidado

Timóteo
2019

Aos meus pais,
que sempre me apoiaram e não mediram esforços para que eu chegasse até aqui.

Agradecimentos

Agradeço a todos aqueles que de alguma forma contribuíram para a realização deste trabalho.

*“Conheça o seu inimigo como a si mesmo
e não precisa temer o resultado
de cem batalhas”
Sun Tzu*

Resumo

A tradução automática é uma subárea do processamento de linguagens naturais que tem como objetivo a tradução automática de textos de uma língua natural de origem para uma outra língua natural de destino. Dentre os diferentes paradigmas para se realizar a tradução automática, a tradução automática baseada em regras apresenta-se como forte candidata para realizar tradução entre línguas com baixa quantidade de *corpus* disponível. O Apertium foi apontado como uma solução de *software* livre e colaborativa para desenvolvimento de sistemas de tradução automática baseada em regras e foi alvo do presente trabalho. Um dos principais obstáculos para o crescimento do Apertium é a ausência de uma interface homem-máquina específica para manutenção de suas bases de conhecimento. Visando superar esses obstáculos, este trabalho propõe um ambiente *web* que permite com que usuários sem conhecimentos específicos de computação possam contribuir com a expansão das bases de conhecimento do Apertium. A arquitetura proposta integra-se à atual forma de desenvolvimento dessas bases de conhecimento e permite com que usuários leigos em computação possam contribuir com a expansão do vocabulário dos dicionários morfológicos do Apertium.

Palavras-chave: tradução automática, tradução automática baseada em regras, Apertium, GUI.

Abstract

Machine translation is a subarea of natural language processing that aims to automatically translate texts from one natural language to another. Among the different paradigms for machine translation, rule-based machine translation is a strong candidate for translation between low-resource languages. Apertium was appointed as a free/open-source and collaborative solution for the development of rule-based machine translation systems and was the focus of the present work. One of the main obstacles to Apertium's growth is the absence of a specific human-machine interface for maintaining its knowledge bases. Aiming to overcome these obstacles, this paper proposes a web environment that allows users without computer-specific knowledge to contribute to the expansion of Apertium's knowledge bases. The proposed architecture integrates with the current form of development of these knowledge bases and allows lay users in computing to contribute to the expansion of the vocabulary of the morphological dictionaries of Apertium.

Keywords: machine translation, rule-based machine translation, Apertium, GUI.

Lista de ilustrações

Figura 1 – Paradigmas de tradução automática	18
Figura 2 – Módulos do Apertium	20
Figura 3 – Sistema de controle de versão distribuído	26
Figura 4 – Git <i>snapshots</i>	26
Figura 5 – Duas <i>branches</i> apontando para o mesmo <i>commit</i>	27
Figura 6 – <i>Branches</i> divergentes	27
Figura 7 – Componentes da arquitetura proposta	32
Figura 8 – Primeiro passo na inserção de um lema	37
Figura 9 – Segundo passo na inserção de um lema	37
Figura 10 – Terceiro passo na inserção de um lema	38
Figura 11 – Fluxo de trabalho proposto para inserção de lemas	39

Lista de tabelas

Tabela 1 – Elementos da definição de paradigmas	22
Tabela 2 – Processo de análise e geração morfológica	22
Tabela 3 – Operações suportadas pelo recurso Official Dictionary	33
Tabela 4 – Operações suportadas pelo recurso MyDictionary	34
Tabela 5 – Operações suportadas pelo recurso Symbol	34
Tabela 6 – Operações suportadas pelo recurso Paradigm	35
Tabela 7 – Operações suportadas pelo recurso Lemma	35

Lista de abreviaturas e siglas

API	Application Programming Interface
CICD	Continuous Integration and Continuous Delivery
DB	Dicionário bilíngue
DM	Dicionário morfológico
EBMT	Exemple-Based Machine Translation
FL	Forma léxica
FS	Forma superficial
GUI	Graphic User Interface
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
LIBRAS	Linguagem Brasileira de Sinais
LN	Linguagem natural
NMT	Neural Machine Translation
PGRG	Plataformas Gerenciadoras de Repositórios Git
PLN	Processamento de linguagem natural
RBMT	Rule-Based Machine Translation
REST	Representational State Transfer
RNA	Rede neural artificial
RTF	Rich Text Format
SMT	Statistical Machine Translation
TA	Tradução automática
URI	Uniform Resource Identifier
XML	Extensible Markup Language
WDM	Web Dictionary Maintenance

Sumário

1	INTRODUÇÃO	13
1.1	Problema	14
1.2	Objetivos	15
1.3	Estrutura do trabalho	15
2	PROCEDIMENTOS METODOLÓGICOS	16
3	REFERENCIAL TEÓRICO	17
3.1	Linguagem natural e tradução	17
3.2	Tradução automática	17
3.3	Apertium	19
3.3.1	Especificações das bases de dados: Dicionários	20
3.3.1.1	Dicionário monolíngue	20
3.3.1.2	Dicionário bilíngue	22
3.3.1.3	Dicionário de regras	23
3.3.1.4	Dicionário de transformações	23
3.3.2	Arquitetura do Apertium	23
3.3.2.1	Deformatador	23
3.3.2.2	Analisador morfológico	24
3.3.2.3	Etiquetador	24
3.3.2.4	Transferência léxica	24
3.3.2.5	Transferência estrutural	24
3.3.2.6	Gerador morfológico	24
3.3.2.7	Pós-gerador	25
3.3.2.8	Reformatador	25
3.4	Git	25
3.4.1	Branch	25
4	ESTADO DA ARTE	28
4.1	Dixtools	28
4.2	Wikitionary	29
4.3	GitLab	29
5	DESENVOLVIMENTO	31
5.1	Arquitetura	31
5.2	Especificação dos recursos da API	33
5.2.1	Official Dictionary	33
5.2.2	My Dictionary	33
5.2.3	Symbol	34

5.2.4	Paradigm	35
5.2.5	Lemma	35
5.3	Protótipo funcional e validação	36
5.3.1	Inserção de lemas	36
5.4	Workflow	38
6	CONSIDERAÇÕES FINAIS	40
6.1	Trabalhos Futuros	40
	REFERÊNCIAS	42
	APÊNDICES	44
	APÊNDICE A – API	45
	APÊNDICE B – PROJETO DE INTERFACES	76

1 Introdução

O processamento de linguagens naturais (PLN) é uma subárea da ciência da computação que tem por objetivo aprender, entender e produzir conteúdos de linguagens naturais. O PLN vem crescendo nos últimos 20 anos tanto como uma área de pesquisa científica quanto uma área de aplicações práticas (HIRSCHBERG; MANNING, 2015). Um dos principais desafios da computação para as próximas décadas é desenvolver sistemas capazes de processar com eficiência as linguagens naturais (OLLER et al., 2007).

Dentro do campo de PLN, destaca-se a tradução automática (TA), que é uma subárea da linguística computacional que tem como objetivo a tradução automática de textos de uma língua natural de origem para uma língua natural de destino, traduzindo todo seu conteúdo de maneira a gerar um resultado inteligível e que preserve suas características, como estilo, coesão e significado (MIRANDA, 2009).

Os sistemas de TA são cada vez mais necessários devido à natureza multilíngue da sociedade globalizada e das complexas redes sociais, organizacionais e comerciais existentes (OLLER et al., 2007). A comunicação entre povos distintos facilitada por meios eletrônicos é uma realidade e com evoluções cada vez mais esperadas. Contudo, os sistemas de tradução mais bem sucedidos têm sido tradicionalmente *softwares* proprietários, os quais utilizam tanto componentes internos quanto bases de dados fechadas.

Alguns sistemas de TA considerados gratuitos estão disponíveis para uso na internet com algumas restrições. Essas ferramentas são distribuídas com propósitos comerciais e não podem ser adaptadas ou aprimoradas para uma finalidade específica (RAMIRÉZ-SÁNCHEZ et al., 2006). Essa forma de distribuição dificulta a inserção de suporte a novos idiomas, que na maioria das vezes é feita objetivando algum retorno financeiro, e sua utilização em outras aplicações.

Hirschberg e Manning (2015) e Stein (2018) ainda mostram que uma das grandes limitações dos sistemas de PLN em geral é que, muitas vezes, eles estão disponíveis apenas para línguas com alta quantidade de *corpus*¹ disponível, como o Inglês, Francês, Espanhol, Alemão e Chinês. Em contrapartida, diversas línguas com baixa quantidade de *corpus* disponível, como Bengali, Indonésio, Punjabi, Cebuano e Swahili, que são faladas e escritas por milhões de pessoas, podem não possuir tais sistemas disponíveis com uma qualidade aceitável.

Dessa forma, surgiram diversos projetos de *software* livre, que buscam na forma livre e colaborativa de desenvolvimento superar tais dificuldades. Dentre esses projetos, destaca-se o Apertium², que é uma plataforma, licenciada sob a *GNU General Public License*³, para desenvolvimento de máquinas de tradução automática baseadas em regras (*Rule-Based Machine*

¹ conjunto de textos escritos e registros orais em uma determinada língua

² <https://www.apertium.org>

³ <http://www.gnu.org/licenses>

Translation ou RBMT) e que será o foco do presente trabalho.

As RBMT dependem de dados linguísticos explícitos, como dicionários morfológicos, dicionários bilíngues, gramáticas e regras de transferência estrutural (TYERS et al., 2010). A base de conhecimento linguístico do Apertium é formada por arquivos XML com formatos bem definidos para diversos pares de línguas e é disponibilizada publicamente na internet através de repositórios Git.

Atualmente, o Apertium oferece suporte a mais de quarenta línguas e o desenvolvimento de novos pares de tradução vem sendo alvo de diversos trabalhos nos últimos anos, como podemos ver em Tyers et al. (2017) o par italiano-sardenho, em Johnson et al. (2017) o par sámi-finlandês e em Klubička, Ramírez-Sánchez e Ljubešić (2016) o par croata-sérvio.

Uma das principais etapas no processo de desenvolvimento de um novo par de tradução consiste na criação dos dicionários morfológicos das línguas envolvidas, que representam o vocabulário englobado pelo tradutor. A expansão desses dicionários consiste basicamente na associação de palavras ao seu respectivo padrão de flexão. Esse padrão de flexão é denominado paradigma e na maior parte das línguas latinas e germânicas, por exemplo, as palavras compartilham apenas um pequeno grupo deles e eles são, em sua grande maioria, definidos por colaboradores especialistas em linguística nos estudos iniciais para a criação do dicionário morfológico da língua em questão.

A tarefa de associar palavras ao seu paradigma, apesar de ser relativamente simples, exige um alto esforço humano devido ao tamanho dos dicionários morfológicos, que podem chegar a ter mais de oitenta mil registros (dentre paradigmas, símbolos e lemas), e à quantidade de palavras presentes nas línguas. Além disso, essa tarefa é atualmente restrita aos usuários que possuem, além dos conhecimentos necessários na língua em questão, conhecimentos específicos de computação. A expansão do vocabulário dos dicionários morfológicos poderia ser realizada com o auxílio de usuários leigos em computação, desde que existisse uma interface homem-máquina adequada para isso.

1.1 Problema

É evidente que dentre as principais limitações das RBMT, de modo geral, estão o alto custo no desenvolvimento de seus recursos linguísticos e na sua extensão para outros idiomas (CASELI, 2017). Um dos principais obstáculos para o crescimento do Apertium é a ausência de uma interface homem-máquina específica para manutenção de seus dicionários morfológicos, de pares e regras de tradução. As poucas alternativas de interface existentes visam usuários especialistas tanto em computação quanto em linguística (MIRANDA, 2009).

O Apertium conta com poucos contribuidores especialistas, sendo que o número de usuários (que são colaboradores em potencial) é da ordem de milhares. A necessidade de conhecimentos específicos em computação, tais como linguagens de marcação (XML), sistemas de controle de versões (Git) e das estruturas internas do Apertium, acabam limitando o número de colaboradores aptos a contribuir e afastando colaboradores em potencial.

Dessa forma, este trabalho propõe um meio para superar tais limitações e permitir que

os usuários com domínio sobre uma língua contribuam com a expansão das bases de conhecimento do Apertium sem precisarem adquirir conhecimentos específicos de computação.

1.2 Objetivos

Este trabalho tem como objetivo principal propor um ambiente *web* colaborativo - o Apertium *Web Dictionary Maintenance* ou Apertium WDM - para o aumento do vocabulário dos dicionários morfológicos do Apertium, a fim de reduzir a quantidade de conhecimento necessário como pré-requisito para uma pessoa se tornar apta a contribuir. Com isso, espera-se aumentar o número de contribuidores e, por conseguinte, a capacidade de tradução do Apertium.

Também, objetivam-se mais especificamente:

1. Especificar uma API RESTful que forneça um conjunto de funcionalidades para adição, de forma colaborativa, de novas palavras aos dicionários morfológicos de línguas de origem latina e germânica do Apertium, que são armazenados em um repositório Git;
2. Especificar uma GUI (*Graphical User Interface*) que permita ao usuário utilizar de forma prática os recursos fornecidos pela API;
3. Propor um fluxo de trabalho colaborativo entre os contribuidores e os mantenedores das bases de dados.

1.3 Estrutura do trabalho

Este trabalho está estruturado da seguinte forma:

- O Capítulo 2 apresenta os procedimentos metodológicos através dos quais esse trabalho se desenvolve.
- As bases teóricas são apresentadas no Capítulo 3, que inclui os principais conceitos para melhor entendimento do trabalho.
- No Capítulo 4 são apresentadas algumas ferramentas para manutenção de dicionários e desenvolvimento colaborativo que são relacionadas a esse trabalho.
- O ambiente proposto como solução ao problema apresentado é descrito no Capítulo 5.
- Por fim, no Capítulo 6 são apresentadas as considerações finais desse trabalho.

2 Procedimentos Metodológicos

Conforme Gerhardt e Silveira (2009), esta pesquisa é caracterizada como qualitativa quanto à sua abordagem; como aplicada quanto à sua natureza; como exploratória quanto aos seus objetivos; e como experimental quanto aos procedimentos utilizados.

Os procedimentos metodológicos foram organizados nas seguintes etapas:

1. estudo das estruturas internas do Apertium;
2. pesquisa por soluções alternativas para manutenção das bases de conhecimentos de tradutores automáticos;
3. levantamento dos requisitos necessários para desenvolvimento de uma interface para gestão de conhecimento;
4. desenvolvimento da arquitetura para solução do problema;
5. desenvolvimento de um protótipo funcional da arquitetura proposta, que envolve a criação de uma API e de uma interface gráfica para utilizá-la;
6. validação do protótipo através de testes de inserção de lemas ao dicionário morfológico de português.

3 Referencial Teórico

Este capítulo aborda os principais conceitos fundamentais para melhor compreensão da aplicabilidade e do contexto ao qual se engloba este trabalho.

3.1 Linguagem natural e tradução

A linguagem pode ser definida como uma ferramenta de comunicação que permite a troca de informações entre dois indivíduos. A linguagem natural (LN) é a forma mais comum de comunicação e origina-se na capacidade inata de comunicação do ser humano. Ela é representada, por exemplo, pelas linguagens faladas, como o português, e pelas linguagens de sinais, como a Linguagem Brasileira de Sinais (Libras).

Atualmente, evidencia-se cada vez mais a necessidade da tradução entre LN devido à globalização e à natureza cada vez mais multicultural da sociedade. Mais de 7000 línguas já foram catalogadas ao redor do mundo. No Brasil, 237 já foram catalogadas, sendo que, destas, 217 estão vivas e 20 já são consideradas extintas (EBERHARD; SIMONS; FENNIG, 2019).

A automatização do processo de tradução vem sendo um dos principais desafios da computação nas últimas décadas. Os maiores empecilhos para uma tradução automática eficiente se devem à natureza dinâmica e ambígua das LN, que são consideradas objetos vivos e que estão em constante mudança.

3.2 Tradução automática

Tradução automática é o processo de tradução automatizado no qual um programa de computador traduz um texto de uma LN para outra.

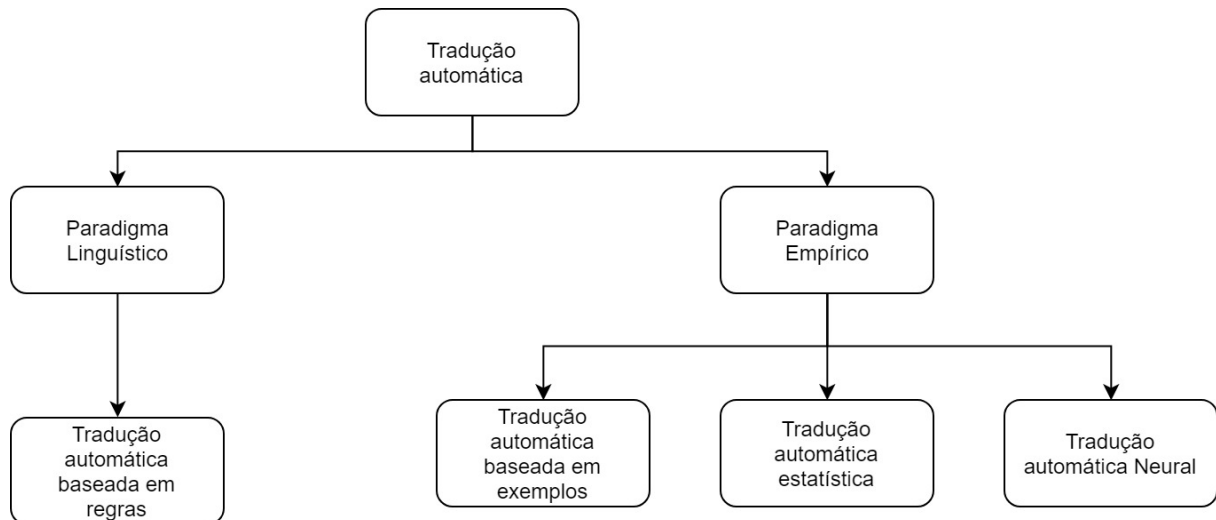
Desde o início dos estudos da TA até os dias atuais, as principais estratégias utilizadas no processo de tradução para gerar um texto na língua alvo são a tradução direta, a tradução por transferência e a tradução por interlíngua (MIRANDA, 2009; CASELI, 2017).

Na tradução direta, as unidades lexicais do texto na língua de origem são diretamente mapeadas para unidades lexicais correspondentes na língua de destino sem que haja nenhum processamento intermediário, sendo considerado o processo mais simples e que produz resultados pouco satisfatórios.

A tradução por transferência, por sua vez, “[...]realiza uma análise sintática parcial ou completa da língua fonte e o mapeamento fonte-alvo se dá com base em regras de transferência sintática[...]” (CASELI, 2017, p. 1783).

Por fim, temos a tradução por interlíngua, que realiza uma representação completa do texto na língua de origem para uma língua intermediária e, então, é gerado o texto na língua alvo a partir do texto na língua intermediária.

Figura 1 – Paradigmas de tradução automática



Fonte: Elaborado pelo autor

Os sistemas de TA são, ainda, classificados quanto ao paradigma utilizado (vide Figura 1), sendo eles: o paradigma linguístico, onde o conhecimento das línguas de origem e de destino são mapeados na forma de regras como ocorre nas RBMT, e o paradigma empírico, onde a tradução é realizado sobre a análise de *corpus* bilíngues e que engloba a tradução automática estatística (*Statistical Machine Translation* ou SMT), a tradução automática baseada em exemplos (*Example-based Machine Translation* ou EBMT) e a recente tradução automática neural (*Neural Machine Translation* ou NMT) (CASELI, 2017).

Esses quatros modelos teóricos de tradução são discutidos a seguir.

- **Tradução automática estatística:** Na SMT as traduções são geradas a partir de modelos estatísticos cujos parâmetros são derivados da análise de *corpus* linguísticos bi-língues. Essa paradigma era considerado o estado da arte e o Google o utilizava em seu tradutor (Google Translator) até 2016, quando deu lugar à NMT (CASELI, 2017). SMT procuram balancear a probabilidade de que a palavra da sentença traduzida corresponda a palavra da sentença original (fidelidade) e que as palavras da sentença traduzida são essas e que ocorrem nessa ordem no idioma alvo (fluência) (FORCADA et al., 2011). Dentre suas principais vantagens estão a produção de uma tradução mais fluente, na maioria das vezes, quando comparado a RBMT (FORCADA et al., 2011) e a independência quanto ao idioma utilizado, uma vez que é possível gerar tradutores para diferentes idiomas utilizando os mesmos algoritmos. Em contrapartida, a SMT não é aplicável caso não exista *corpus* suficientes para a língua em questão, a qualidade da tradução pode ser imprevisível e existe uma dificuldade em interpretar e modificar os valores gerados pelo modelo estatístico (CASELI, 2017).
- **Tradução automática baseada em exemplos:** Assim como a SMT, a EBMT utiliza a análise de *corpus* linguísticos bilíngues para realizar suas traduções. Porém, ao contrário da SMT, que de forma resumida busca combinar palavras ou frases com maior probabili-

dade, a EBMT busca encontrar padrões de tradução nos fragmentos apresentados como exemplos.

- **Tradução automática baseada em regras:** Por sua vez, as RBMT abordam o paradigma baseado em informações linguísticas e contam com inúmeras regras de tradução incorporadas a diversos dicionários disponíveis para cada par de tradução. As traduções são geradas a partir de regras morfológicas, sintáticas e semânticas além de regras de transferência de um idioma para outro. A criação manual das regras é, ao mesmo tempo, uma das principais vantagens e desvantagens dessa abordagem. A possibilidade de definição explícita de cada regra permite ao projetista do tradutor tratar de forma eficiente casos específicos de tradução, mas em contrapartida, a criação de todas as regras de tradução demandam um grande esforço humano.
- **Tradução automática neural:** A tradução utilizando redes neurais artificiais (RNA), por fim, é uma nova abordagem que surgiu nos últimos anos que objetiva o treinamento de uma RNA para a tradução entre dois idiomas. Ao contrário dos outros métodos de tradução, que são subdivididos em diversos componentes, esse método visa a construção e treinamento de uma única e grande RNA que lê uma sentença e gera sua tradução (BAHDANAU; CHO; BENGIO, 2015). As principais vantagens em relação aos modelos tradicionais são os poucos conhecimentos linguísticos necessários para gerar o tradutor, a otimização conjunta de toda a rede e não apenas de módulos específicos e o fato de que o conhecimento gerado é mais compacto, do que, por exemplo, os dicionários das RBMT (CASELI, 2017). Contudo, esse modelo costuma ser menos preciso que os modelos estatísticos (até então considerado estado da arte). Isso ocorre por três motivos principais: o processo de treinamento e de tradução são lentos; é ineficaz em lidar com palavras raras; e algumas vezes falha ao traduzir todas as palavras de uma sentença (WU et al., 2016). Esses problemas têm impedindo que a tradução utilizando RNA seja utilizada em serviços e implementações práticas, onde a velocidade e precisão são essenciais (WU et al., 2016).

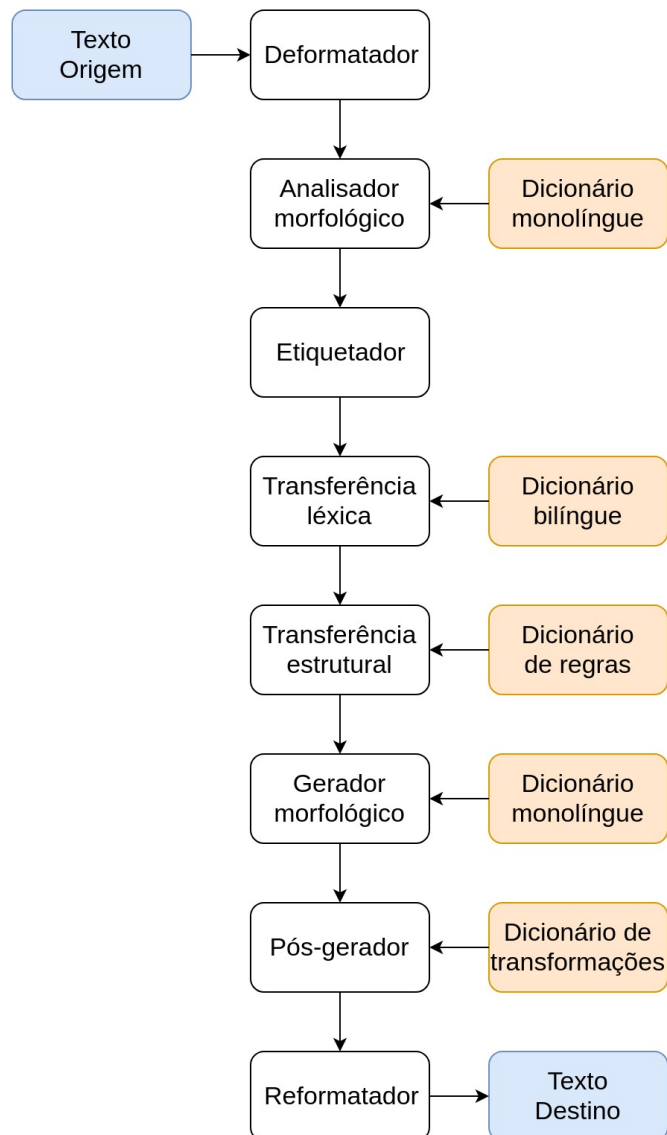
3.3 Apertium

O Apertium é uma plataforma de tradução resultante do projeto “Máquina de Tradução Automática Open-Source para as línguas da Espanha” (“*Traducción automática de código abierto para las lenguas del estado español*”) desenvolvido na Universidade de Alicante e financiado pelo governo espanhol e catalão, a fim de criar uma MTA entre as línguas da Espanha. Ele, o Apertium, é um *shallow-transfer rule-based machine translation*¹ desenvolvido, inicialmente, para traduzir idiomas semelhantes. Com o passar dos anos, notou-se a eficiência de sua tradução e a partir de então sua base de conhecimento vem sendo expandida para dezenas de outros pares de línguas distantes.

O mecanismo de tradução do Apertium é composto por 8 módulos e utiliza quatro dicionários de dados que estão representados na Figura 2 e que estão descritos nas subseções

¹ Máquina de tradução automática baseada em regras de transferências superficiais

Figura 2 – Módulos do Apertium



Fonte: Forcada et al. (2010). Adaptado pelo autor.

seguintes.

3.3.1 Especificações das bases de dados: Dicionários

3.3.1.1 Dicionário monolíngue

Os dicionários monolíngues (ou dicionários morfológicos ou ainda DM) são utilizados no processo de análise morfológica e também no processo de geração morfológica. Eles são compostos pela definição de um alfabeto e de um conjunto de símbolos, paradigmas e lemas.

- **Definição do alfabeto:** Consiste em definir o conjunto de caracteres que são utilizados pela língua. O alfabeto é utilizado na etapa de análise morfológica e permite ao analisador gerar corretamente os *tokens* das palavras do texto de entrada. Sua definição é

delimitada pelo elemento *<alphabet>*, como no Exemplo 3.1 que mostra a definição do alfabeto utilizado no DM de português.

Exemplo 3.1 – Alfabeto do dicionário morfológico de português

```
<alphabet>
  ÁÀÁÁÇÊÊÊÊÏÏÏÏÎÑÕÕÕÕÛÛÛÛÜäääääçèèèèííííîñðóóóóôúúúABCDEF GHI JKLMNOPQ
  RSTUVWXYZabcdefghijklmnopqrstuvwxyz
</alphabet>
```

- **Definição dos símbolos:** Consiste na definição das unidades gramaticais utilizados pelo dicionário. A definição dos símbolos é delimitada pelo elemento *<sdefs>* e cada símbolo é representado pelo elemento *<sdef>*. O atributo “n” define o nome do símbolo. O Exemplo 3.2 mostra a definição de alguns símbolos utilizados no DM de português sendo eles: substantivos (n, do inglês *noun*), nomes pessoais (np), adjetivos (adj), feminino (f), masculino (m), singular (sg), plural (pl) e advérbio (adv).

Exemplo 3.2 – Símbolos do dicionário morfológico de português

```
<sdefs>
  <sdef n="n"/>
  <sdef n="np"/>
  <sdef n="adj"/>
  <sdef n="f"/>
  <sdef n="m"/>
  <sdef n="sg"/>
  <sdef n="pl"/>
  <sdef n="adv"/>
  ...
</sdefs>
```

- **Definição dos paradigmas:** Em grande parte das línguas, muitos lemas compartilham o mesmo padrão de flexão. Utiliza-se então paradigmas, que são padrões de flexões comuns à várias palavras, para agrupá-las e evitar-se, então, escrever todas flexões de todas palavras. A definição dos paradigmas é delimitada pelo elemento *<pardefs>* e cada paradigma é representado por um elemento *<pardef>*.

O Exemplo 3.3 mostra a definição do paradigma “abadia” do dicionário de português, que é utilizado por 6968 lemas.

Exemplo 3.3 – Paradigmas do dicionário morfológico de português

```
<pardefs>
  ...
  <pardef n="abadia_n">
    <e><p><l>s</l><r><s n="n"/><s n="f"/><s n="pl"/></r>
    </p></e>
    <e><p><l></l><r><s n="n"/><s n="f"/><s n="sg"/></r></p>
    </e>
  </pardef>
  ...
</pardefs>
```

A Tabela 1 descreve os demais elementos utilizados na definição de um paradigma. É adotado a abordagem de pares de direção, pois posteriormente o dicionário será compilado para uma máquina de estados finitos e será utilizado tanto no processo de análise

morfológica (direção esquerda para direita) quanto no processo de geração morfológica (direção direita para esquerda), como descrito nas seções 3.3.2.2 e 3.3.2.6.

Elemento	Descrição
e	Define o início de uma nova entrada
p	Define o início de novo par de direções
l	Do inglês <i>left</i> , define a entrada esquerda
r	Do inglês <i>right</i> , define a entrada direita
s	Define o símbolo associado à entrada direita

Tabela 1 – Elementos da definição de paradigmas

A Tabela 2 mostra o processo de análise e geração morfológica utilizando o paradigma definido no Exemplo 3.3.

Entrada	Direção	Saída
abadia	esquerda para direita	abadia<n><f><sg>
abadias	esquerda para direita	abadia<n><f><pl>
abadia<n><f><sg>	direita para esquerda	abadia
abadia<n><f><pl>	direita para esquerda	abadias

Tabela 2 – Processo de análise e geração morfológica

- **Definição dos lemas:** Nessa seção ocorre a definição de todos os lemas presentes na língua e suas respectivas associações com o paradigma utilizado. Eles são agrupados em seções que são delimitadas pelo elemento <section>. O Exemplo 3.4 mostra a definição de alguns lemas do DM de português. Cada lema é representado pelo elemento <e>. O atributo “lm” define o nome que representa esse lema e os elementos <i> e <par> definem, respectivamente, a parte invariante do lema e o paradigma associado a ele.

Exemplo 3.4 – Lemas do dicionário morfológico de português

```
<section id="main" type="standard">
  ...
  <e lm="contingência"><i>contingência</i><par n="abadia__n"
/></e>
  <e lm="advertência"><i>advertência</i><par n="abadia__n"/>
</e>
  <e lm="adaptabilidade"><i>adaptabilidade</i><par n="abadi
a__n"/></e>
  ...
  <e lm="fraternizar"><i>fraterniz</i><par n="abal/ar__vble
x"/></e>
  <e lm="reciclar"><i>recicl</i><par n="abal/ar__vblex"/></
e>
  ...
</section>
```

3.3.1.2 Dicionário bilíngue

Os dicionários bilíngues (DB) armazenam os mapeamentos de tradução de cada lema de uma língua para outra. Eles são compostos pela definição dos símbolos utilizados nas duas

línguas e por uma única seção que realiza a associação dos lemas.

O Exemplo 3.5 mostra como seria a estrutura de um dicionário bilíngue português-espanhol. Dentro da seção principal, é dado o exemplo de associação do lema em português “menino” ao seu correspondente “*niño*” em espanhol.

Exemplo 3.5 – Exemplo dicionário bilíngue português-espanhol

```
<dictionary>
  <sdefs>
    ...
    <sdef n="n"/>
    <sdef n="np"/>
    <sdef n="adj"/>
    <sdef n="f"/>
    <sdef n="m"/>
    <sdef n="mf"/>
    <sdef n="sg"/>
    <sdef n="pl"/>
    <sdef n="adv"/>
    ...
  </sdefs>
  <section id="main" type="standard">
    ...
    <e><p><l>menino<s n="n"/></l><r>niño<s n="n"/></r></p></e>
    ...
  </section>
</dictionary>
```

3.3.1.3 Dicionário de regras

Os dicionários de regras armazenam as regras de transferência necessárias para a tradução, sendo necessário um dicionário para cada sentido de tradução. As regras são formadas por padrões de sequência em que categorias lexicais aparecem no texto de entrada (FORCADA et al., 2010). Ao serem detectados, os ajustes definidos por essa regra serão aplicados no texto de saída. Esse dicionário não será abordado neste trabalho.

3.3.1.4 Dicionário de transformações

Os dicionários de transformações armazenam as regras para modificação das palavras e é utilizado após a etapa de geração morfológica. Tais modificações são, por exemplo, contrações das palavras, acréscimo de apóstrofe e outras modificações que uma palavra pode sofrer caso haja uma outra palavra específica próxima à ela. Assim como o dicionário de regras, esse dicionário também não será abordado neste trabalho.

3.3.2 Arquitetura do Apertium

3.3.2.1 Deformatador

O deformatador é o módulo responsável por separar o texto a ser traduzido das informações de formato (HTML, RTF, etc.) em que ele se encontra (FORCADA et al., 2010). O Exemplo 3.6 mostra um texto em HTML a ser deformatado.

Exemplo 3.6 – Texto de entrada do deformatador

```
<header> Texto a ser traduzido </header>
```

O Exemplo 3.7 mostra o texto deformatado.

Exemplo 3.7 – Texto de saída do deformatador

```
[<header>] Texto a ser traduzido [</header>]
```

Todas as informações de marcação do formato são colocadas entre colchetes que serão ignorados pelos módulos subsequentes.

3.3.2.2 Analisador morfológico

O analisador morfológico, como o próprio nome já diz, é responsável por realizar a análise morfológica das palavras na forma superficial (FS), ou seja, a forma em que a palavra aparece no texto e gerar seus correspondentes na forma léxica (FL). A forma léxica, por sua vez, é composta pelo lema da palavra e por etiquetas que representam sua categoria (substantivo, adjetivo, pronome, etc.) e suas flexões (gênero, número, tempo verbal, etc.). Cada palavra pode possuir mais de uma forma léxica devido à ambiguidade presente nas LN.

3.3.2.3 Etiquetador

O módulo de etiquetagem é responsável por escolher a FL correta dentre as possíveis alternativas geradas pelo módulo anterior (devido a ambiguidade supracitada). Para tal, ele utiliza um modelo estatístico (mais especificamente o modelo oculto de Markov) (FORCADA et al., 2010).

3.3.2.4 Transferência léxica

Na etapa de transferência léxica é onde ocorre, de fato, a tradução. Nesse módulo é utilizado o dicionário bilíngue e realizado o mapeamento de cada FL do idioma de origem para o idioma de destino. Os lemas de cada FL são trocados pelos lemas correspondentes no idioma alvo e as etiquetas são mantidas (exceto quando exista explicitamente uma indicação para mudá-las).

3.3.2.5 Transferência estrutural

O módulo de transferência estrutural é responsável por processar pequenos trechos ou frases que precisam de um processamento extra devido à divergências gramaticais entre os dois idiomas (mudanças de gênero e número, reordenação de palavras, mudanças de preposições, etc.) (FORCADA et al., 2010). Para tanto, é utilizado o dicionário de regras que diz qual ação deve ser tomada para cada padrão encontrado.

3.3.2.6 Gerador morfológico

O gerador morfológico, por sua vez, utiliza o dicionário monolíngue da língua alvo para gerar as palavras do texto na forma superficial a partir da forma léxica.

3.3.2.7 Pós-gerador

O módulo pós-gerador utiliza um dicionário de transformações para, por exemplo, criar abreviações e contrações no texto traduzido.

3.3.2.8 Reformatador

O reformatador é responsável por converter o texto traduzido para o formatado de entrada, mantendo sua formatação original.

3.4 Git

Git é uma ferramenta distribuída de controle de versões, originalmente projetada para desenvolvimento de *software*, mas que também pode ser utilizada para registrar o histórico de edição de qualquer arquivo de texto.

A ferramenta teve origem em 2005, quando Linus Torvalds viu a necessidade de um sistema de controle de versões que fosse rápido, com suporte a desenvolvimento não linear, completamente distribuído e capaz de lidar com projetos grandes para ser utilizado no *kernel* do Linux.

Os sistemas de controle de versão distribuído utilizam o paradigma cliente-servidor. Entretanto, todos os clientes mantêm uma versão completa do repositório e de todo seu histórico de modificações. Caso ocorra uma falha no servidor, qualquer cliente pode restaurar seus dados (CHACON; STRAUB, 2014). A Figura 3 mostra essa arquitetura.

O funcionamento do Git consiste em basicamente na realização e armazenamento de *snapshots*² do projeto. Sempre que o usuário realiza um *commit*³, uma cópia de todos os arquivos do projeto é salva e associada a um *snapshot*. Caso um arquivo não tenha sido alterado, por questões de eficiência, é salvo apenas uma referência para o último *snapshot* em que esse arquivo foi alterado (CHACON; STRAUB, 2014). A Figura 4 exemplifica esse processo, onde, em cada versão, os arquivos que estão tracejados não foram modificados e são apenas uma referência para a última versão onde eles foram, de fato, modificados.

Para garantir a integridade, um *checksum* de todos os arquivos é realizado antes de cada *commit* ser armazenado e um identificador único nesse repositório (uma *hash* SHA-1) é gerado e associado a ele (CHACON; STRAUB, 2014). Outras informações, como nome e *e-mail* do autor e data e hora da criação do *commit*, também são armazenadas.

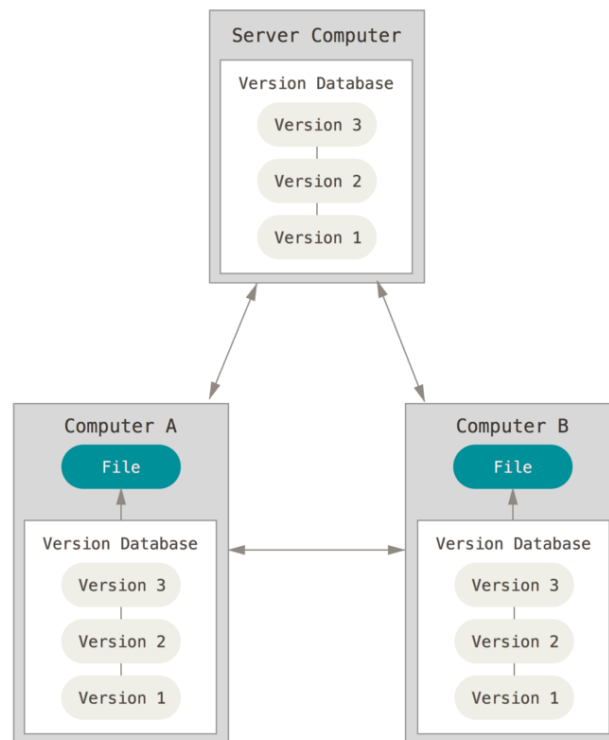
3.4.1 Branch

A utilização de *branches* (ou ramificações) é a forma de desenvolvimento não linear adotada pelo Git. Em cada *commit* é salvo, além das informações mencionadas anteriormente, um ponteiro (ou referência) para o *commit* anterior, exceto para o primeiro *commit* realizado

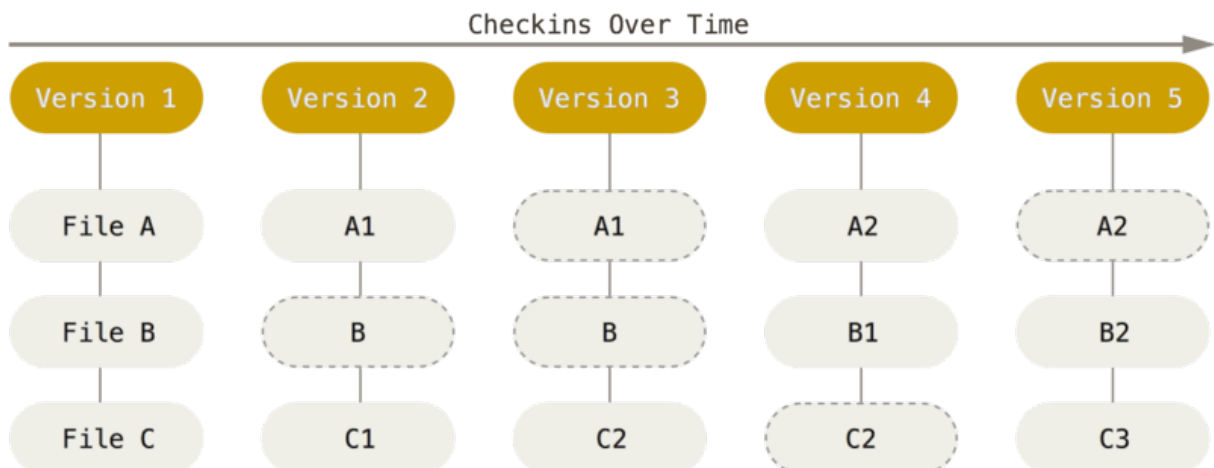
² *Snapshot*: retrato dos arquivos do sistema em um dado momento.

³ *Commit*: salvar o estado atual do projeto. Muitas vezes utilizado como sinônimo de *snapshot*.

Figura 3 – Sistema de controle de versão distribuído



Fonte: Extraído de Chacon e Straub (2014).

Figura 4 – Git *snapshots*

Fonte: Extraído de Chacon e Straub (2014).

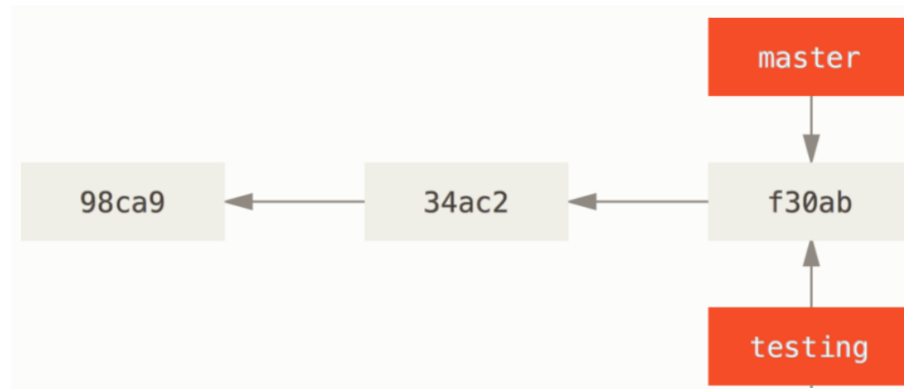
e para *commits* originizados da junção de duas ou mais *branches*, que armazenarão, então, um ponteiro para cada uma delas.

Todo repositório Git é inicializado com uma *branch* principal que é denominada *master*. Uma *branch* é um ponteiro para um *commit* que se move automaticamente para o *commit* recém adicionado.

Iniciar uma nova *branch* consiste, então, em criar um novo ponteiro que irá se mover

de forma independente dos demais. A Figura 5 mostra um repositório com três *commits* (em cinza) e duas *branches* (em vermelho), sendo elas a *branch* principal (*master*) e uma nova *branch* criada, que foi chamada de “testing”.

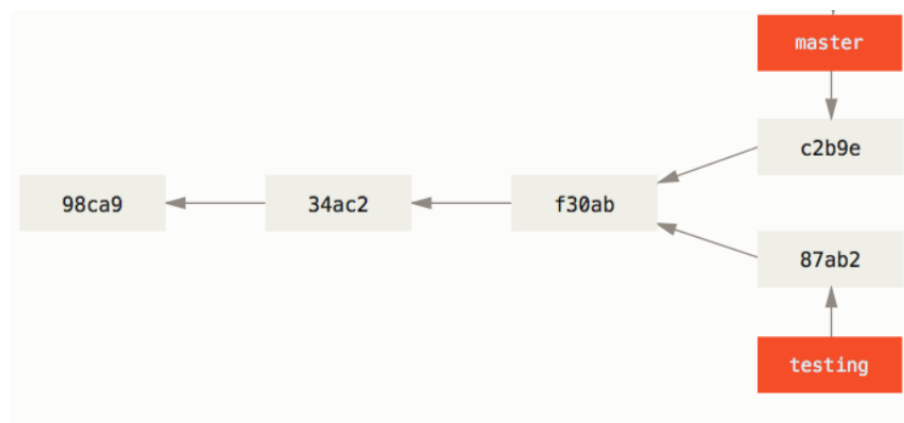
Figura 5 – Duas *branches* apontando para o mesmo *commit*



Fonte: Extraído de Chacon e Straub (2014).

Cada uma dessas *branches* representa uma “linha do tempo” de desenvolvimento independente. A Figura 6 mostra essas duas *branches* após a realização de um *commit* em cada uma delas.

Figura 6 – *Branches* divergentes



Fonte: Extraído de Chacon e Straub (2014).

4 Estado Da Arte

Atualmente, o desenvolvimento colaborativo das bases de conhecimento do Apertium se dá através de contribuições de usuários com domínio sobre a língua em questão e que possuem também conhecimentos específicos de computação. Os contribuidores realizam, na maioria das vezes, a manipulação dos extensos arquivos XML dos dicionários através de editores de texto genéricos, sem utilização de nenhuma ferramenta com interface gráfica específica para o conteúdo dos dicionários. As contribuições são, então, enviadas para os repositórios Git dos dicionários através de outras ferramentas com esse fim.

A seguir, são apresentadas as principais ferramentas de auxílio na manutenção de dicionários morfológicos e no desenvolvimento colaborativo.

4.1 Dixtools

O Apertium-dixtools¹ é uma ferramenta de código fonte livre, disponível em linha de comando, que dispõe algumas funcionalidades para manipulação dos dicionários do Apertium. Dentre elas, destacam-se:

- **Formatar dicionário:** reestrutura as definições de palavras e suas traduções, formatando, por exemplo, uma palavra por linha.
- **Ordernar um dicionário:** ordena as palavras em ordem alfabética.
- **Leitor de dicionário:** obtém listas dos elementos (lemas, paradigmas, definições...) de um dicionário.
- **Inverter dicionário:** cria um dicionário bilíngue de uma língua A para uma língua B a partir de um dicionário bilíngue da língua B para língua A.
- **Crossdics:** constrói um dicionário bilíngue de uma linguagem A para outra linguagem C a partir de um dicionário da linguagem A para linguagem B e de um dicionário da linguagem B para a linguagem C.
- **Cobertura do dicionário:** gera dados estatístico da frequência de uso de diferentes palavras do dicionário.
- **Autoconcord:** faz dicionários bilíngues entrarem em concordância com os dicionários monolíngues quando existem diferenças (gênero, número, etc.) entre as duas linguagens.
- **Paradigmas equivalentes:** utilizado para encontrar paradigmas não utilizados e paradigmas funcionando da mesma forma que outro.
- **Unir dicionários:** permite unir lista de palavras de vários dicionários monolíngues.

¹ <http://wiki.apertium.org/wiki/Apertium-dixtools>

- **Aumentar:** permite adicionar novas palavras ao dicionário interativamente.
- **Grep:** permite filtrar um dicionário baseado em um lema ou paradigma específico e gerar um novo dicionário.

Apesar de fornecer funcionalidades úteis, a instalação e utilização do Dixtools é feita apenas através de linha de comando, o que acaba dificultando sua utilização por pessoas leigas em computação. Além disso, ele permite apenas que os dicionários sejam modificados localmente, sem nenhum tipo de colaboração, sendo para isso necessário a utilização de outras ferramentas externas.

Parte do código fonte do Dixtools que realiza a representação e relacionamento dos elementos de um dicionário, bem como sua leitura e escrita, será reutilizado neste trabalho.

4.2 Wikitionary

O Wikitionary² é um projeto *web* multilíngue pertencente a Wikimedia Foundation³ que tem como objetivo criar um dicionário eletrônico de conteúdo livre. Ele possui mais de 170 linguagens e é desenvolvido colaborativamente por voluntários da comunidade através do *software* Wiki, o que permite que quase todas as pessoas com acesso ao site possam modificar seu conteúdo.

Contudo, as palavras que o compõem o Wikitionary podem não possuir todas as classificações necessárias para serem diretamente exportadas e utilizadas em um dicionário do Apertium. Entretanto, elas podem vir a serem utilizadas como sugestões de novas palavras que ainda não compõem o vocabulário de um dicionário do Apertium para que os colaboradores possam acrescentá-las.

4.3 GitLab

GitLab⁴ é uma plataforma de *software* livre para gerenciamento de repositórios Git, licenciada sob a licença MIT, que implementa algumas funcionalidades extras, como o suporte a Wiki, o gerenciamento de tarefas e o CI/CD. Ele dispõe de uma interface gráfica que permite aos usuários visualizarem estatísticas e gerenciarem seus repositórios Git de forma prática e eficiente. Além disso, dispõe uma API que permite que outras aplicações realizem operações sobre os repositórios de forma automatizada.

Todas as bases de conhecimento do Apertium (dicionários) são armazenadas em repositórios Git que mantêm toda linha do tempo de desenvolvimento e serve com aparato para

² <https://www.wiktionary.org>

³ <https://wikimediafoundation.org>

⁴ <https://about.gitlab.com/>

o desenvolvimento colaborativo. O ambiente *web* proposto estará conectado ao GitLab através de sua API e ele será usado como ferramenta administrativa e de controle de versão dos dicionários.

5 Desenvolvimento

Este capítulo descreve a arquitetura do ambiente *web* colaborativo proposto como solução ao problema apresentado.

5.1 Arquitetura

O ambiente proposto tem como principal filosofia proporcionar um meio pelo qual usuários não especialistas em computação possam contribuir com a expansão do vocabulário de um dicionário morfológico. Em discussões com um pequeno grupo de desenvolvedores de dicionários do Apertium, foram levantados os seguintes requisitos, de natureza exploratória, que um sistema com esse fim deveria atender para ser bem-sucedido:

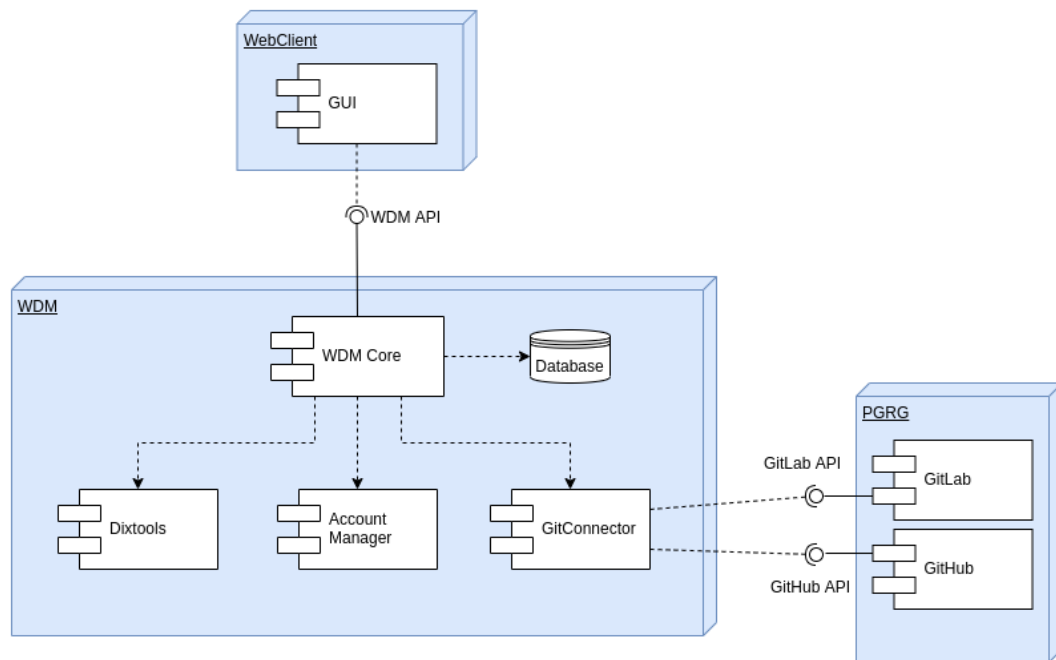
- A estrutura dos dicionários não deve ser alterada;
- A arquitetura deve se integrar à atual forma de desenvolvimento colaborativo;
- O histórico de contribuições deve ser preservado;
- Deve ser possível identificar o autor de cada contribuição;
- A arquitetura deve ser incremental, sendo possível expandi-la para o gerenciamento dos demais elementos dos dicionários morfológicos e também para o gerenciamento dos demais dicionários.

Dessa forma, a Figura 7 apresenta os componentes da arquitetura proposta que contempla tais requisitos. Para melhor compreensão, a arquitetura foi dividida em três módulos que são descritos na sequência.

O PGRG (Plataformas Gerenciadoras de Repositórios Git) é um módulo externo ao projeto e é constituído por plataformas que realizam o gerenciamento de repositórios Git, tais como o GitLab e o GitHub. Essas plataformas são responsáveis por gerenciarem os repositórios que armazenam os dicionários e os disponibilizarem publicamente na internet. Para serem utilizadas nessa arquitetura, elas devem prover uma API que permita a realização das operações básicas sobre esses repositórios, tais como a criação de *commits*, *branches*, *download* de arquivos, etc.

Como apresentado na Seção 3.3.1, os dicionários são arquivos de texto extensos no formato XML que muitas vezes chegam a ter mais de 100 mil linhas de conteúdo. Esse tamanho demasiado acarreta em dificuldades na manipulação desses arquivos. Dessa forma, o módulo *Web Client* é responsável por disponibilizar uma GUI que permita com que os usuários realizem modificações pontuais sobre esses arquivos de textos de forma facilitada, organizada e sem efeitos colaterais.

Figura 7 – Componentes da arquitetura proposta



Fonte: Elaborado pelo autor

Para tanto, a GUI utiliza os recursos de uma API disponibilizada pelo módulo WDM. Esse módulo, por sua vez, é responsável por carregar uma versão do dicionário do repositório Git em memória, realizar a manipulação desse dicionário localmente, de forma que a sua integridade seja mantida, e devolver o arquivo em forma de texto para seu repositório com as contribuições do usuário. Os componentes que compõem o módulo WDM são descritos a seguir.

- **WDM Core:** O WDM Core é o principal componente do módulo WDM. Ele é responsável por realizar a integração entre os outros componentes, além de processar as requisições recebidas através da API que é especificada na Seção 5.2.
- **Dixtools:** O componente Dixtools é responsável por manipular diretamente os arquivos XML, realizando as operações de inserção, remoção e modificação dos elementos dos dicionários. Como apresentado na Seção 4.1, esse componente é uma adaptação da ferramenta (de mesmo nome) disponibilizada pela comunidade do Apertium.
- **GitConnector:** O componente GitConnector é responsável por realizar a comunicação com as plataformas de gerenciamento de repositórios Git (tal como o GitLab e o GitHub). Ele depende diretamente da API fornecida por essas plataformas. Para este trabalho, utilizou-se a plataforma GitLab por atender aos requisitos necessários e apresentar-se como uma solução de *software* livre. Outras plataformas podem ser utilizadas através da expansão desse componente.
- **Account Manager:** Por fim, o componente Account Manager é responsável pelo registro

e autenticação de usuários. Através dele, é possível identificar o autor de cada contribuição.

5.2 Especificação dos recursos da API

Nessa seção são especificados os recursos da API provida pelo módulo WDM e são mostrados alguns exemplos de resposta das chamadas. A documentação completa da API pode ser encontrada no Apêndice A.

Os recursos da API seguem ao padrão de uma API RESTful. Foram utilizadas as operações POST, GET, PUT e DELETE. Os recursos são identificados através da URI (*Uniform Resource Identifier*) e as repostas utilizam o formato JSON (*JavaScript Object Notation*).

5.2.1 Official Dictionary

O recurso *Official Dictionary* é utilizado para listagem dos dicionários oficiais e para sua sincronização com os repositórios Git. Cada dicionário oficial possui um repositório distinto e é representado por sua *branch* principal. Diferentes versões do dicionário podem ser utilizadas na plataforma, sendo que para isso as versões devem estar identificadas por *tags* na *branch* principal. A Tabela 3 mostra a URI relativa desse recurso e as operações que ele suporta. O Exemplo 5.1 mostra o JSON de resposta da chamada para obter todos os dicionários disponíveis.

URI	POST	GET	PUT	DELETE
/officialdictionary		X		
/officialdictionary/{officialDictionaryId}		X		
/officialdictionary/synchronize	X			

Tabela 3 – Operações suportadas pelo recurso Official Dictionary

Exemplo 5.1 – Exemplo de resposta do recurso Official Dictionary

```
[
  {
    "id": number,
    "language": "string",
    "reference": "string",
    "version": "string"
  }
]
```

5.2.2 My Dictionary

O recurso *My Dictionary* permite ao usuário criar sua cópia de um dicionário oficial. Para isso, ele deve indicar o dicionário oficial e a versão desejada e, então, uma nova *branch* exclusiva será criada no repositório do dicionário oficial e armazenará as modificações feitas pelo usuário. O usuário poderá, então, realizar suas alterações nessa *branch* criada exclusivamente para ele e posteriormente criar uma solicitação de *merge* com a *branch* principal para

enviar suas contribuições para a comunidade ou realizar o *download* do arquivo XML do dicionário para alimentar seu próprio tradutor. A Tabela 4 mostra a URI relativa desse recurso e as operações que ele suporta. O Exemplo 5.2 mostra o JSON de resposta da chamada para obter todos os dicionários do usuário.

URI	POST	GET	PUT	DELETE
/mydictionary	X	X		
/mydictionary/{myDictionaryId}		X	X	X
/mydictionary/{myDictionaryId}/merge	X			
/mydictionary/{myDictionaryId}/download		X		

Tabela 4 – Operações suportadas pelo recurso MyDictionary

Exemplo 5.2 – Exemplo de resposta do recurso My Dictionary

```
[
  {
    "id": number,
    "name": "string",
    "officialDictionaryId": number,
    "qtLemma": number,
    "qtParadigm": number,
    "qtSymbol": number
  }
]
```

5.2.3 Symbol

O recurso *Symbol* é utilizado para realizar operações sobre os símbolos de um *My Dictionary*. A única operação permitida é a de leitura, sendo a criação e edição, bem como a remoção de símbolos, não abordadas no contexto deste trabalho. A Tabela 5 mostra a URI relativa desse recurso e as operações que ele suporta. O Exemplo 5.3 mostra o JSON de resposta da chama para obter todos os símbolos de um *My Dictionary*.

URI	POST	GET	PUT	DELETE
/symbol/{myDictionaryId}		X		
/symbol/{myDictionaryId}/{name}		X		

Tabela 5 – Operações suportadas pelo recurso Symbol

Exemplo 5.3 – Exemplo de resposta do recurso Symbol

```
[
  {
    "name": "string",
    "comment": "string"
  }
]
```

Nos dicionários oficiais, os símbolos aparecem de forma abreviada seguindo uma nomenclatura comum à maioria das línguas. Para uma maior praticidade e facilidade para o

usuário final, foi adicionado o campo “*comment*” às respostas do recurso *Symbol* para mostrar o nome do símbolo de forma estendida e/ou sua descrição, baseado no glossário¹ presente na *Wiki* do Apertium.

5.2.4 Paradigm

O recurso *Paradigm* é utilizado para realizar operações sobre os paradigmas de um dicionário. Assim como o recurso *Symbol*, a única operação suportada no contexto deste trabalho é a de leitura. A Tabela 6 mostra a URI relativa desse recurso e as operações que ele suporta. O Exemplo 5.4 mostra o JSON de resposta da chama para obter todos os paradigmas de um *My Dictionary*.

URI	POST	GET	PUT	DELETE
/paradigm/{myDictionaryId}		X		
/paradigm/{myDictionaryId}/{name}		X		
/paradigm/{myDictionaryId}/suggestion		X		

Tabela 6 – Operações suportadas pelo recurso Paradigm

Exemplo 5.4 – Exemplo de resposta do recurso Paradigma

```
[
  {
    "name": "string",
    "entries": [
      {
        "left": "string",
        "right": "string",
      }
    ]
  }
]
```

5.2.5 Lemma

O recurso *Lemma* é utilizado para realizar operações sobre os lemas do dicionário. A Tabela 7 mostra a URI relativa desse recurso e as operações que ele suporta. O Exemplo 5.5 mostra o JSON de resposta da chama para obter todos os lemas de um *My Dictionary*.

URI	POST	GET	PUT	DELETE
/lemma/{myDictionaryId}	X	X		
/lemma/{myDictionaryId}/{lineNo}		X	X	X

Tabela 7 – Operações suportadas pelo recurso Lemma

¹ http://wiki.apertium.org/wiki/List_of_symbols

Exemplo 5.5 – Exemplo de resposta do recurso Lemma

```
[
  {
    "name": "string",
    "root": "string",
    "paradigm": "string",
    "lineNo": number,
    "inflections": [
      "value": "string",
      "symbols": [
        "string"
      ]
    ]
  }
]
```

5.3 Protótipo funcional e validação

Foi desenvolvido um protótipo funcional para testar os recursos da API proposta e simular a inserção de lemas em um dicionário morfológico. O projeto de interface completo pode ser encontrado no Apêndice B. O desenvolvimento do protótipo envolveu:

- Cadastro e *login* de usuários;
- Listagem e sincronização de *Official Dictionaries* com os repositórios Git;
- Gerenciamento de *My Dictionaries*;
- Listagem de símbolos;
- Listagem de paradigmas;
- Gerenciamento de lemas.

5.3.1 Inserção de lemas

A operação de inserção de um lema foi dividida em três passos. A Figura 8 apresenta o primeiro desses passos, que consiste na definição do nome do lema e de sua parte invariante (*root*), seguido de duas informações opcionais: a classe gramatical do lema e alguns exemplos de flexão, que serão utilizados para encontrar sugestões de possíveis paradigmas para esse lema.

Foi utilizado como exemplo a inserção do lema “gato” em um dicionário morfológico de português. As informações inseridas foram: sua parte invariante, o prefixo “gat”; sua classe gramatical, “*noun*” (substantivo); e duas de suas possíveis flexões: “gato” e “gatas”.

No segundo passo, apresentado na Figura 9, ocorre a sugestão dos possíveis paradigmas ao usuário. Quanto mais precisas forem as informações apresentadas na etapa anterior, menor será a quantidade de paradigmas sugeridos. O usuário deverá, então, verificar qual das sugestões se encaixa ao lema. Na tabela inferior dessa imagem, são apresentadas todas

Figura 8 – Primeiro passo na inserção de um lema

Apertium WDM

Home Entities Administration Account

← Cancel

Step 1 Step 2 Step 3

Lemma informations

Name

gato

Root

gat

Part of speech

Noun

Inflections

gato x gatos x

→ Next

Fonte: Elaborado pelo autor

as flexões e seus respectivos símbolos do lema em questão, utilizando o paradigma sugerido selecionado na tabela superior.

Figura 9 – Segundo passo na inserção de um lema

Apertium WDM

Home Entities Administration Account

← Cancel

Step 1 Step 2 Step 3

Paradigm suggestions

Paradigm

adept/o_n

afegã/o_n

1

Details

Inflection	Symbols
gatas	<noun><feminine><plural>
gata	<noun><feminine><singular>
gatos	<noun><masculine><plural>
gato	<noun><masculine><singular>

1

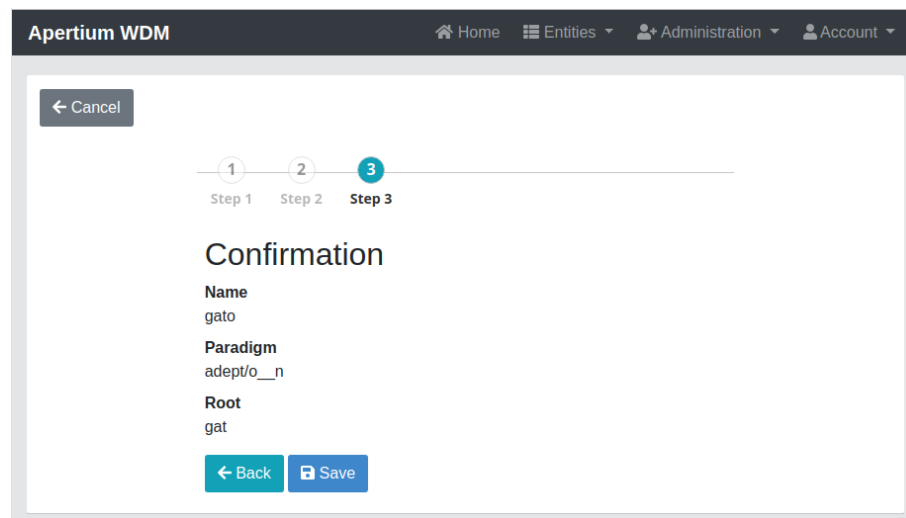
← Back → Next

Fonte: Elaborado pelo autor

No exemplo utilizado, as informações inseridas no primeiro passo resultaram na sugestão de apenas dois paradigmas, denominados “adept/o__n” e “afegã/o__n”. Ao selecionar cada um deles, são mostradas todas as flexões do lema “gato” utilizando-se esse paradigma. Cabe ao usuário verificar se cada flexão condiz com seus símbolos e escolher a sugestão correta.

Na terceira e última etapa, apresentada na Figura 10, o usuário deve apenas confirmar as informações inseridas.

Figura 10 – Terceiro passo na inserção de um lema



Fonte: Elaborado pelo autor

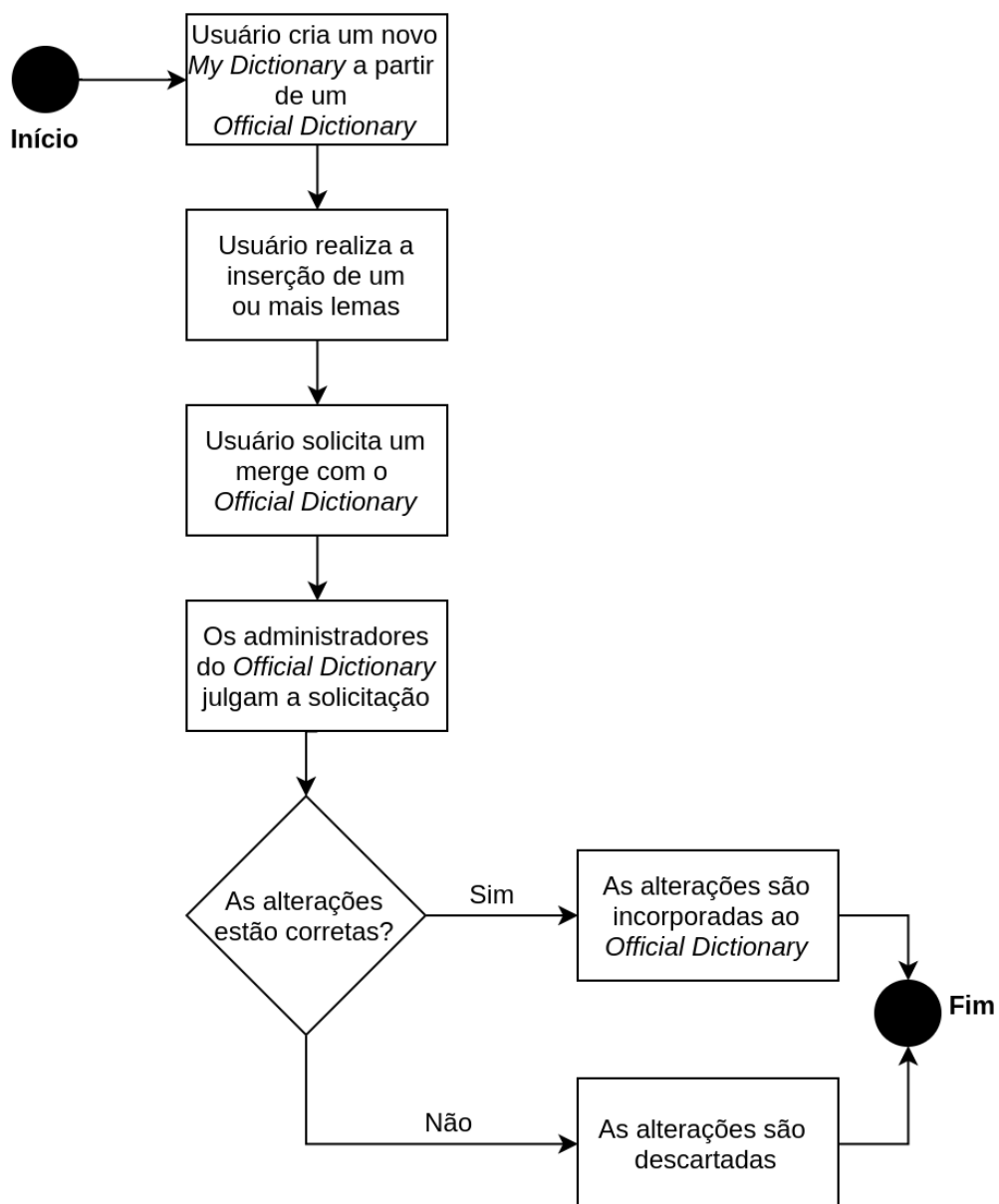
Ao fim desse processo, o lema é inserido no *My Dictionary* em questão. Em outras palavras, é criado um *commit* na *branch* exclusiva do usuário no repositório Git do dicionário em questão contendo essa modificação. A partir de então, o usuário pode realizar operações como solicitação de *merge* com o dicionário oficial ou *download* do arquivo XML do dicionário, como descrito na Seção 5.2.2.

5.4 Workflow

A arquitetura apresentada se integra a atual forma de desenvolvimento utilizando-se, por exemplo, o fluxo de trabalho para inserção de um novo lema em um dicionário morfológico apresentado na Figura 11.

Através do Apertium WDM, o usuário realiza as três primeiras etapas apresentadas, que consistem na criação um novo dicionário para o usuário (uma *branch* no repositório Git); na inserção de lemas (*commits* individuais na *branch* exclusiva do dicionário criado); e por fim no envio das contribuições para avaliação (*merge request* entre a *branch* criada e a *branch* principal). As etapas subsequentes já estão presentes no fluxo de trabalho atual, onde os mantenedores das bases de dados recebem as contribuições através de *merge requests* e verificam se as alterações devem ser aceitas ou não.

Figura 11 – Fluxo de trabalho proposto para inserção de lemas



Fonte: Elaborado pelo autor

6 Considerações Finais

Neste trabalho, foi realizado um estudo bibliográfico sobre a tradução automática. Nessa área, a tradução automática baseada em regras foi apontada como uma forte alternativa para realização de tradução automática entre línguas com baixa quantidade de *corpus* disponível.

O Apertium foi apresentado como uma das principais soluções de *software* livre para o desenvolvimento colaborativo de sistemas de tradução automática baseada em regras e foi exposto que dentre suas principais limitações está o alto custo no desenvolvimento de seus recursos linguísticos.

Visando superar as dificuldades do desenvolvimento desses recursos linguísticos, foi proposto um ambiente *web* colaborativo que permite com que usuários não especialistas em computação possam contribuir com a expansão dos dicionários morfológicos utilizados no Apertium, de forma integrada à sua atual forma de desenvolvimento e sem precisarem adquirir conhecimentos específicos de computação.

O protótipo funcional desenvolvido contempla apenas as operações essenciais para adição de novos lemas aos dicionários morfológicos. Entretanto, a arquitetura pode ser facilmente expandida, de forma que permita realizar um conjunto maior de operações sobre os demais objetos desse dicionário e até mesmo para outros tipos de dicionário, como o bilíngue.

Dessa forma, o objetivo geral deste trabalho foi alcançado, uma vez que a arquitetura proposta mostrou-se como uma boa alternativa para abstrair os conhecimentos em computação necessários para uma pessoa se tornar apta a contribuir com a expansão dos dicionários morfológicos do Apertium. Ademais, o presente trabalho emerge como uma passo inicial para o desenvolvimento de um ambiente específico mais abrangente para desenvolvimento colaborativo e gerenciamento completo das bases de conhecimento do Apertium, além de permitir que outros desenvolvedores criem módulos editores para essas bases de conhecimento do Apertium utilizando os recursos disponibilizados pela API proposta.

6.1 Trabalhos Futuros

Devido à complexidade envolvida no desenvolvimento desse ambiente, bem como a complexidade da estrutura dos dicionários, este trabalho focou no desenvolvimento das operações básicas de representação dos elementos dos dicionários morfológicos, na inserção de lemas e na integração da utilização da ferramenta com a atual forma desenvolvimento. Como sequência desse trabalho, recomenda-se:

- propor uma interface gráfica adequada para criação de paradigmas de dicionários morfológicos;

- realizar testes de desempenho, comparando a manipulação dos dicionários em memória e a representação dos dicionários de forma indexada em um banco de dados;
- expandir o ambiente para oferecer suporte à manipulação dos outros dicionários, como o bilíngue;
- avaliar o uso da plataforma por usuários-finais;
- integrar ferramentas de testes de cobertura e de erros dos dicionários ao ambiente proposto.

Referências

- BAHDANAU, D.; CHO, K.; BENGIO, Y. *NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE*. 2015. Citado na página 19.
- CASELI, H. de M. Tradução automática: estratégias e limitações. 12 2017. Citado nas páginas 14, 17, 18 e 19.
- CHACON, S.; STRAUB, B. *Pro Git*. 2nd. ed. Berkely, CA, USA: Apress, 2014. ISBN 1484200772, 9781484200773. Citado nas páginas 25, 26 e 27.
- EBERHARD, D. M.; SIMONS, G. F.; FENNIG, C. D. *Ethnologue: Languages of the World*. 2019. Disponível em: <<https://www.ethnologue.com>>. Citado na página 17.
- FORCADA, M. L. et al. *Documentation of the Open-Source Shallow-Transfer Machine Translation Platform Apertium*. 2010. Citado nas páginas 20, 23 e 24.
- FORCADA, M. L. et al. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, v. 25, n. 2, p. 127–144, Jun 2011. ISSN 1573-0573. Disponível em: <<https://doi.org/10.1007/s10590-011-9090-0>>. Citado na página 18.
- GERHARDT, T.; SILVEIRA, D. *Métodos de Pesquisa*. [S.l.: s.n.], 2009. (Série Educação a Distância - UFRGS). ISBN 9788538600718. Citado na página 16.
- HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. *Science*, v. 349, n. 6245, p. 261–266, 2015. Citado na página 13.
- JOHNSON, R. et al. *North Sámi to Finnish rule-based machine translation system*. 2017. Citado na página 14.
- KLUBIČKA, F.; RAMÍREZ-SÁNCHEZ, G.; LJUBEŠIĆ, N. Collaborative development of a rule-based machine translator between croatian and serbian. In: *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*. [S.l.: s.n.], 2016. p. 361–367. Citado na página 14.
- MIRANDA, A. *WiKLaTS – UM AMBIENTE DE INTERFACE E INTERAÇÃO PARA MANIPULAÇÃO E FORMALIZAÇÃO DE CONHECIMENTO*. 2009. 80 p. — Universidade Federal do Paraná, Curitiba, 2009. Citado nas páginas 13, 14 e 17.
- OLLER, C. A. et al. *Apertium, una plataforma de código abierto para el desarrollo de sistemas de traducción automática*. 2007. Citado na página 13.
- RAMIRÉZ-SÁNCHEZ, G. et al. Opentrad apertium open-source machine translation system: an opportunity for business and research. 2006. Citado na página 13.
- STEIN, D. *Machine translation: Past, present and future*. [s.n.], 2018. 5–17 p. ISBN 9783946234739. Disponível em: <<http://langsci-press.org/catalog/book/106>>. Citado na página 13.
- TYERS, F. et al. Free/open-source resources in the apertium platform for machine translation research and development. *The Prague Bulletin of Mathematical Linguistics*, v. 93, 01 2010. Citado na página 14.

TYERS, F. M. et al. *Rule-Based Machine Translation for the Italian–Sardinian Language Pair*. 2017. Citado na página 14.

WU, Y. et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. Citado na página 19.

Apêndices

APÊNDICE A – API

WDM API

Base URL: /, **Version:** 0.0.1

WDM API documentation

Schemes:

Summary

Tag: Account

Operations about Account

Operation	Description
GET /api/account	Get an Account
POST /api/account	Update an Account
POST /api/account/change-password	Change Account's password
POST /api/account/reset-password/finish	Finish Account's password reset
POST /api/account/reset-password/init	Reset Account's password
GET /api/activate	Active an Account
GET /api/authenticate	Check if an Account is authenticated
POST /api/register	Create an Account

Tag: Lemma

Operations about Lemma

Operation	Description
GET /api/lemma/{myDictionaryID}	Get all lemmas from a My Dictionary
POST /api/lemma/{myDictionaryID}	Create a lemma
GET /api/lemma/{myDictionaryID}/{lineno}	Get lemma by Line No.
PUT /api/lemma/{myDictionaryID}/{lineno}	Update a lemma
DELETE /api/lemma/{myDictionaryID}/{lineno}	Delete a lemma

Tag: My Dictionary

Operations about My Dictionary

Operation	Description
GET /api/myDictionary	Get all My Dictionaries
POST /api/myDictionary	Create a My Dictionary

Operation	Description
PUT /api/myDictionary	Update a My Dictionary
GET /api/myDictionary/{id}	Get a My Dictionary by ID
DELETE /api/myDictionary/{id}	Delete a My Dictionary by ID
POST /api/myDictionary/{id}/merge	Create a merge request

Tag: Official Dictionary

Operations about Official Dictionary

Operation	Description
GET /api/officialDictionary	Get all Official Dictionaries
POST /api/officialDictionary/sync	Synchronizes the database Official Dictionaries with the Git repository
GET /api/officialDictionary/{id}	Get Official Dictionary by ID

Tag: Paradigm

Operations about Paradigm

Operation	Description
GET /api/paradigm/{myDictionaryID}	Get all paradigms
GET /api/paradigm/{myDictionaryID}/find	Get paradigm by name
GET /api/paradigm/{myDictionaryID}/suggestion	Get a list of paradigm suggestions

Tag: Symbol

Operations about Symbol

Operation	Description
GET /api/symbol/{myDictionaryID}	Get all symbols from a My Dictionary
GET /api/symbol/{myDictionaryID}/{symbolName}	Get a symbol from a My Dictionary by name

Tag: User

Operations about User

Operation	Description
GET /api/users	Get all Users
POST /api/users	Create an User
PUT /api/users	Update an User
GET /api/users/authorities	Get User's authorities
GET /api/users/{login}	Get an User by login

Operation	Description
DELETE /api/users/{login}	Delete an User by login

Tag: UserAuth

User authentication resource

Operation	Description
POST /api/authenticate	Authenticate a user

Paths

GET /api/account
Tags: Account

Get an Account

DESCRIPTION

RESPONSES

/

200 OK
OK

UserDTO

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

POST /api/account
Tags: Account

Update an Account

DESCRIPTION

REQUEST BODY

application/json

userDTO

UserDTO

RESPONSES

/

200 OK
OK

201 Created
Created

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

POST /api/account/change-password

Change Account's password

Tags: Account

DESCRIPTION

REQUEST BODY

application/json

passwordChangeDto

PasswordChangeDTO

RESPONSES

/

200 OK
OK

201 Created
Created

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

POST /api/account/reset-password/finish

Tags: Account

Finish Account's password reset

DESCRIPTION

REQUEST BODY

application/json

keyAndPassword

KeyAndPasswordVM

RESPONSES

/

200 OK

OK

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/account/reset-password/init

Tags: Account

Reset Account's password

DESCRIPTION

REQUEST BODY

application/json

mail

RESPONSES

/

200 OK

OK

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/activate

Active an Account

Tags: Account

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
key	key	query	string	required

RESPONSES

/

200 OK

OK

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/authenticate

Check if an Account is authenticated

Tags: Account

DESCRIPTION

RESPONSES

/

200 OK

OK

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/authenticate

Authenticate a user

Tags: UserAuth

DESCRIPTION

REQUEST BODY

application/json

loginVM

LoginVM

RESPONSES

/

200 OK

OK

JWTToken

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/lemma/{myDictionaryID}

Get all lemmas from a My Dictionary

Tags: Lemma

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required
page	Page number of the requested page	query	<i>integer</i> (int32)	
size	Size of a page	query	<i>integer</i> (int32)	
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	<i>string[]</i> , multiple parameters (sort=aaa&sort=bbb)	

RESPONSES



200 OK
OK

ITEMS

LemmaDTO

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

POST /api/lemma/{myDictionaryID}

Create a lemma

Tags: Lemma

DESCRIPTION

REQUEST BODY

application/json

lemma

LemmaCreateDTO

REQUEST PARAMETERS

Name	Description	Type	Data type
myDictionaryID	myDictionaryID	path	integer (int64) required

RESPONSES

/

200 OK
OK

LemmaDTO

201 Created
Created

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

DELETE /api/lemma/{myDictionaryID}/{lineno}

Delete a lemma

Tags: Lemma

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
------	-------------	------	-----------

Name	Description	Type	Data type	
lineno	lineno	path	<i>integer</i> (int32)	required
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required

RESPONSES

200 OK
OK

204 No Content
No Content

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

GET /api/lemma/{myDictionaryID}/{lineno}

Get lemma by Line No.

Tags: Lemma

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
lineno	lineno	path	integer (int32)	required
myDictionaryID	myDictionaryID	path	integer (int64)	required

RESPONSES

/

200 OK

OK

LemmaDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

PUT /api/lemma/{myDictionaryID}/{lineno}

Update a lemma

Tags: Lemma

DESCRIPTION

REQUEST BODY

application/json

lemma

LemmaCreateDTO

REQUEST PARAMETERS

Name	Description	Type	Data type	
lineno	lineno	path	<i>integer</i> (int32)	required
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required

RESPONSES

/

200 OK

OK

LemmaDTO

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/myDictionary

Tags: My Dictionary

Get all My Dictionaries

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
page	Page number of the requested page	query	integer (int32)
size	Size of a page	query	integer (int32)
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	string[] , multiple parameters (sort=aaa&sort=bbb)

RESPONSES

200 OK

OK

ITEMS

MyDictionaryDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/myDictionary

Tags: My Dictionary

Create a My Dictionary

DESCRIPTION

REQUEST BODY

application/json

myDictionaryDTO

MyDictionaryDTO

RESPONSES

/

200 OK

OK

MyDictionaryDTO

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

PUT /api/myDictionary

Update a My Dictionary

Tags: My Dictionary

DESCRIPTION

REQUEST BODY

application/json

myDictionaryDTO

MyDictionaryDTO

RESPONSES

/

200 OK

OK

MyDictionaryDTO

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

DELETE /api/myDictionary/{id}

Tags: My Dictionary

Delete a My Dictionary by ID

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	id	path	integer (int64)	required

RESPONSES

/

200 OK

OK

204 No Content

No Content

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

GET /api/myDictionary/{id}

Tags: My Dictionary

Get a My Dictionary by ID

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	id	path	<i>integer</i> (int64)	required

RESPONSES

200 OK

OK

MyDictionaryDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/myDictionary/{id}/merge

Create a merge request

Tags: My Dictionary

DESCRIPTION

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	id	path	<i>integer</i> (int64)	required

RESPONSES

200 OK

OK

ResponseEntity

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/officialDictionary

Get all Official Dictionaries

Tags: Official Dictionary

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
page	Page number of the requested page	query	<i>integer</i> (int32)
size	Size of a page	query	<i>integer</i> (int32)
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	<i>string[]</i> , multiple parameters (sort=aaa&sort=bbb)

RESPONSES



200 OK

OK

ITEMS

OfficialDictionaryDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST
/api/officialDictionary/sync

Synchronizes the database Official Dictionaries with the Git repository

Tags: Official Dictionary

DESCRIPTION

REQUEST BODY

```
application/json
```

RESPONSES

**

200 OK

OK

ResponseEntity

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/officialDictionary/{id}

Get Official Dictionary by ID

Tags: Official Dictionary

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
id	id	path	<i>integer</i> (int64) required

RESPONSES

**

200 OK

OK

OfficialDictionaryDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/paradigm/{myDictionaryID}

Get all paradigms

Tags: Paradigm

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required
page	Page number of the requested page	query	<i>integer</i> (int32)	
size	Size of a page	query	<i>integer</i> (int32)	
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	<i>string[]</i> , multiple parameters (sort=aaa&sort=bbb)	

RESPONSES

/

200 OK

OK

ITEMS

ParadigmDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/paradigm/{myDictionaryID}/find

Get paradigm by name

Tags: Paradigm

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required
name	name	query	<i>string</i>	required

RESPONSES

200 OK

OK

ParadigmDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/paradigm/{myDictionaryID}/suggestion

Get a list of paradigm suggestions

Tags: Paradigm

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
inflections		query	<i>string</i> [] , multiple parameters (inflections=aaa&inflections=bbb)
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64) required
pos		query	<i>string</i>
root		query	<i>string</i>

RESPONSES

200 OK

ITEMS

SuggestionResponseDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/register Tags: Account	Create an Account
DESCRIPTION	
REQUEST BODY application/json managedUserVM ManagedUserVM	
RESPONSES 200 OK	

201 Created
Created

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

GET /api/symbol/{myDictionaryID} Get all symbols from a My Dictionary
Tags: Symbol

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required
page	Page number of the requested page	query	<i>integer</i> (int32)	
size	Size of a page	query	<i>integer</i> (int32)	
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	<i>string[]</i> , multiple parameters (sort=aaa&sort=bbb)	

RESPONSES

200 OK

OK

ITEMS

SymbolDTO

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

GET /api/symbol/{myDictionaryID}/{symbolName} Get a symbol from a My Dictionary by name
Tags: Symbol

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
myDictionaryID	myDictionaryID	path	<i>integer</i> (int64)	required
symbolName	symbolName	path	<i>string</i>	required

RESPONSES

200 OK

OK

SymbolDTO

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

GET /api/users Get all Users
Tags: User

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type
------	-------------	------	-----------

Name	Description	Type	Data type
page	Page number of the requested page	query	<i>integer</i> (int32)
size	Size of a page	query	<i>integer</i> (int32)
sort	Sorting criteria in the format: property(,asc desc). Default sort order is ascending. Multiple sort criteria are supported.	query	<i>string[]</i> , multiple parameters (sort=aaa&sort=bbb)

RESPONSES

200 OK

OK

ITEMS

UserDTO

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

POST /api/users Tags: User	Create an User
DESCRIPTION	
REQUEST BODY application/json userDTO UserDTO	
RESPONSES	

200 OK

OK

User

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

PUT /api/users

Update an User

Tags: User

DESCRIPTION

REQUEST BODY

application/json

userDTO

UserDTO

RESPONSES

200 OK

OK

UserDTO

201 Created

Created

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

GET /api/users/authorities

Get User's authorities

Tags: User

DESCRIPTION

RESPONSES

/

200 OK

OK

ITEMS

string

401 Unauthorized

Unauthorized

403 Forbidden

Forbidden

404 Not Found

Not Found

DELETE /api/users/{login}

Delete an User by login

Tags: User

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
login	login	path	<i>string</i>	required

RESPONSES

/

200 OK

OK

204 No Content

No Content

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

GET /api/users/{login}
Tags: User

Get an User by login

DESCRIPTION

REQUEST PARAMETERS

Name	Description	Type	Data type	
login	login	path	string	required

RESPONSES

/

200 OK
OK

UserDTO

401 Unauthorized
Unauthorized

403 Forbidden
Forbidden

404 Not Found
Not Found

Schema definitions

EntryDTO: object

PROPERTIES

left: string

right: string

InflectionDTO: *object*

PROPERTIES

symbols: *string[]*

ITEMS

string

value: *string*

JWTToken: *object*

PROPERTIES

id_token: *string*

KeyAndPasswordVM: *object*

PROPERTIES

key: *string*

newPassword: *string*

LemmaCreateDTO: *object*

PROPERTIES

name: *string*

paradigm: *string*

root: *string*

LemmaDTO: *object*

PROPERTIES

inflections: *object[]*

ITEMS

InflectionDTO

lineNo: *integer (int32)*

name: *string*

paradigm: *string*

root: *string*

LoginVM: *object*

PROPERTIES

password: *string (4 to 100 chars)* required

rememberMe: *boolean*

username: *string (1 to 50 chars)* required

ManagedUserVM: *object*

PROPERTIES

activated: *boolean*

authorities: *string[]*

ITEMS

string

createdBy: *string*

createdDate: *string (date-time)*

email: *string (5 to 254 chars)*

firstName: *string (up to 50 chars)*

id: *integer (int64)*

imageUrl: *string (up to 256 chars)*

langKey: *string (2 to 10 chars)*

lastModifiedBy: *string*

lastModifiedDate: *string (date-time)*

lastName: *string (up to 50 chars)*

login: *string (1 to 50 chars)* , must match */^[_].@A-Za-z0-9-]*\$/i*

password: *string (4 to 100 chars)*

MyDictionaryDTO: *object*

PROPERTIES

id: *integer (int64)*

name: *string* required

officialDictionaryId: *integer (int64)*

officialDictionaryLanguage: *string*

qtLemma: *integer* (int32)

qtParadigm: *integer* (int32)

qtSymbol: *integer* (int32)

OfficialDictionaryDTO: *object*

PROPERTIES

id: *integer* (int64)

language: *string* required

reference: *string* required

version: *string* required

ParadigmDTO: *object*

PROPERTIES

entries: *object*[]

ITEMS

EntryDTO

name: *string*

PasswordChangeDTO: *object*

PROPERTIES

currentPassword: *string*

newPassword: *string*

ResponseEntity: *object*

PROPERTIES

body: *object*

statusCode: *string*, $x \in \{ "100 CONTINUE", "101 SWITCHING_PROTOCOLS", "102 PROCESSING", "103 CHECKPOINT", "200 OK", "201 CREATED", "202 ACCEPTED", "203 NON_AUTHORITATIVE_INFORMATION", "204 NO_CONTENT", "205 RESET_CONTENT", "206 PARTIAL_CONTENT", "207 MULTI_STATUS", "208 ALREADY_REPORTED", "226 IM_USED", "300 MULTIPLE_CHOICES", "301 MOVED_PERMANENTLY", "302 FOUND", "302 MOVED_TEMPORARILY", "303 SEE_OTHER", "304 NOT_MODIFIED", "305 USE_PROXY", "307 TEMPORARY_REDIRECT", "308$

PERMANENT_REDIRECT", "400 BAD_REQUEST", "401 UNAUTHORIZED", "402 PAYMENT_REQUIRED", "403 FORBIDDEN", "404 NOT_FOUND", "405 METHOD_NOT_ALLOWED", "406 NOT_ACCEPTABLE", "407 PROXY_AUTHENTICATION_REQUIRED", "408 REQUEST_TIMEOUT", "409 CONFLICT", "410 GONE", "411 LENGTH_REQUIRED", "412 PRECONDITION_FAILED", "413 PAYLOAD_TOO_LARGE", "413 REQUEST_ENTITY_TOO_LARGE", "414 URI_TOO_LONG", "414 REQUEST_URI_TOO_LONG", "415 UNSUPPORTED_MEDIA_TYPE", "416 REQUESTED_RANGE_NOT_SATISFIABLE", "417 EXPECTATION_FAILED", "418 I_AM_A_TEAPOT", "419 INSUFFICIENT_SPACE_ON_RESOURCE", "420 METHOD_FAILURE", "421 DESTINATION_LOCKED", "422 UNPROCESSABLE_ENTITY", "423 LOCKED", "424 FAILED_DEPENDENCY", "426 UPGRADE_REQUIRED", "428 PRECONDITION_REQUIRED", "429 TOO_MANY_REQUESTS", "431 REQUEST_HEADER_FIELDS_TOO_LARGE", "451 UNAVAILABLE_FOR_LEGAL_REASONS", "500 INTERNAL_SERVER_ERROR", "501 NOT_IMPLEMENTED", "502 BAD_GATEWAY", "503 SERVICE_UNAVAILABLE", "504 GATEWAY_TIMEOUT", "505 HTTP_VERSION_NOT_SUPPORTED", "506 VARIANT_ALSO_NEGOTIATES", "507 INSUFFICIENT_STORAGE", "508 LOOP_DETECTED", "509 BANDWIDTH_LIMIT_EXCEEDED", "510 NOT_EXTENDED", "511 NETWORK_AUTHENTICATION_REQUIRED" }

statusCodeValue: *integer (int32)*

SuggestionResponseDTO: *object*

PROPERTIES

inflections: *string[]*

ITEMS

string

paradigm: *string*

SymbolDTO: *object*

PROPERTIES

comment: *string*

name: *string*

User: *object*

PROPERTIES

activated: *boolean* required

email: *string* (5 to 254 chars)

firstName: *string* (up to 50 chars)

id: *integer (int64)*

imageUrl: *string* (up to 256 chars)

langKey: *string* (2 to 10 chars)

lastName: *string* (up to 50 chars)

login: *string* (1 to 50 chars) , must match `/^[_@A-Za-z0-9-]*$/` required

resetDate: *string* (date-time)

UserDTO: *object*

PROPERTIES

activated: *boolean*

authorities: *string[]*

ITEMS

string

createdBy: *string*

createdDate: *string* (date-time)

email: *string* (5 to 254 chars)

firstName: *string* (up to 50 chars)

id: *integer* (int64)

imageUrl: *string* (up to 256 chars)

langKey: *string* (2 to 10 chars)

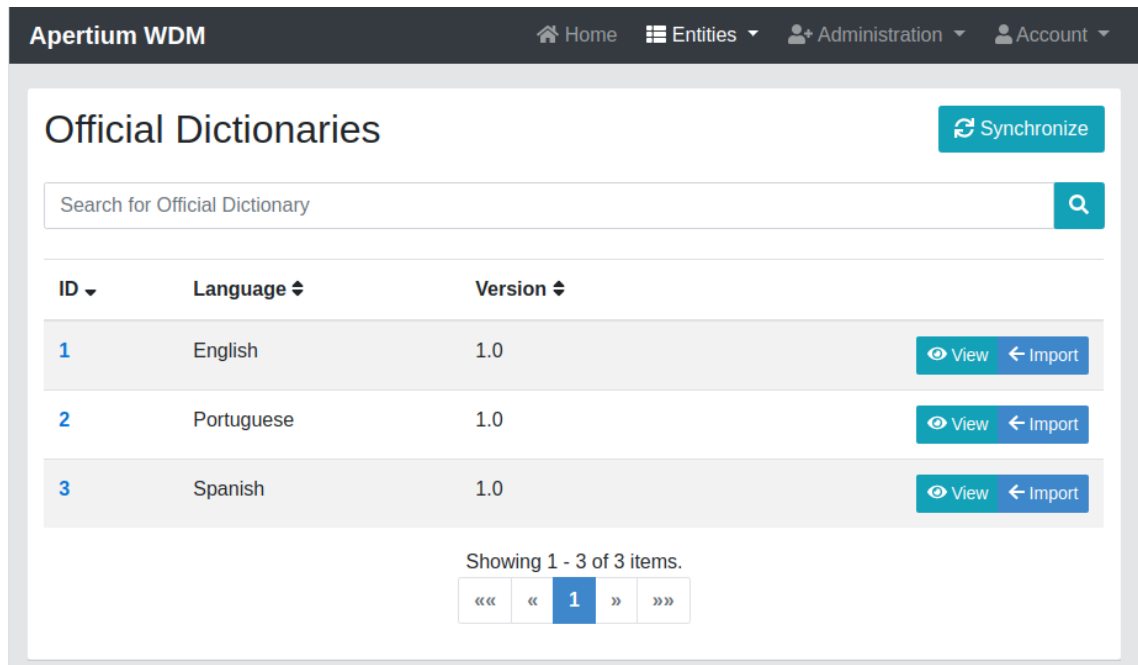
lastModifiedBy: *string*

lastModifiedDate: *string* (date-time)

lastName: *string* (up to 50 chars)

login: *string* (1 to 50 chars) , must match `/^[_@A-Za-z0-9-]*$/`

APÊNDICE B – Projeto de interfaces



Tela de *Official Dictionaries*

Apertium WDM Home Entities Administration Account

My Dictionaries

[+ Create a new My Dictionary](#)

Search for My Dictionary

ID	Name	Qt Paradigm	Qt Lemma	Qt Symbol	Official Dictionary	
1	Dicionário_POR	347	81032	77	Portuguese	Merge Download View Edit Delete
2	Dicionário_ENG	392	62294	86	English	Merge Download View Edit Delete
3	Dicionário_ESP	484	72750	70	Spanish	Merge Download View Edit Delete

Showing 1 - 3 of 3 items.

«« « 1 » »»

Tela de *My Dictionaries*

Apertium WDM Home Entities Administration Account

My Dictionary 1

Name
Dicionário_POR

Paradigms [View](#)
347

Lemmas [View](#) [+ Create a new Lemma](#)
81032

Symbols [View](#)
77

Official Dictionary
[Portuguese](#)

[← Back](#) [Edit](#)

Tela de detalhe de *My Dictionary*

Apertium WDMHomeEntitiesAdministrationAccount

Symbols

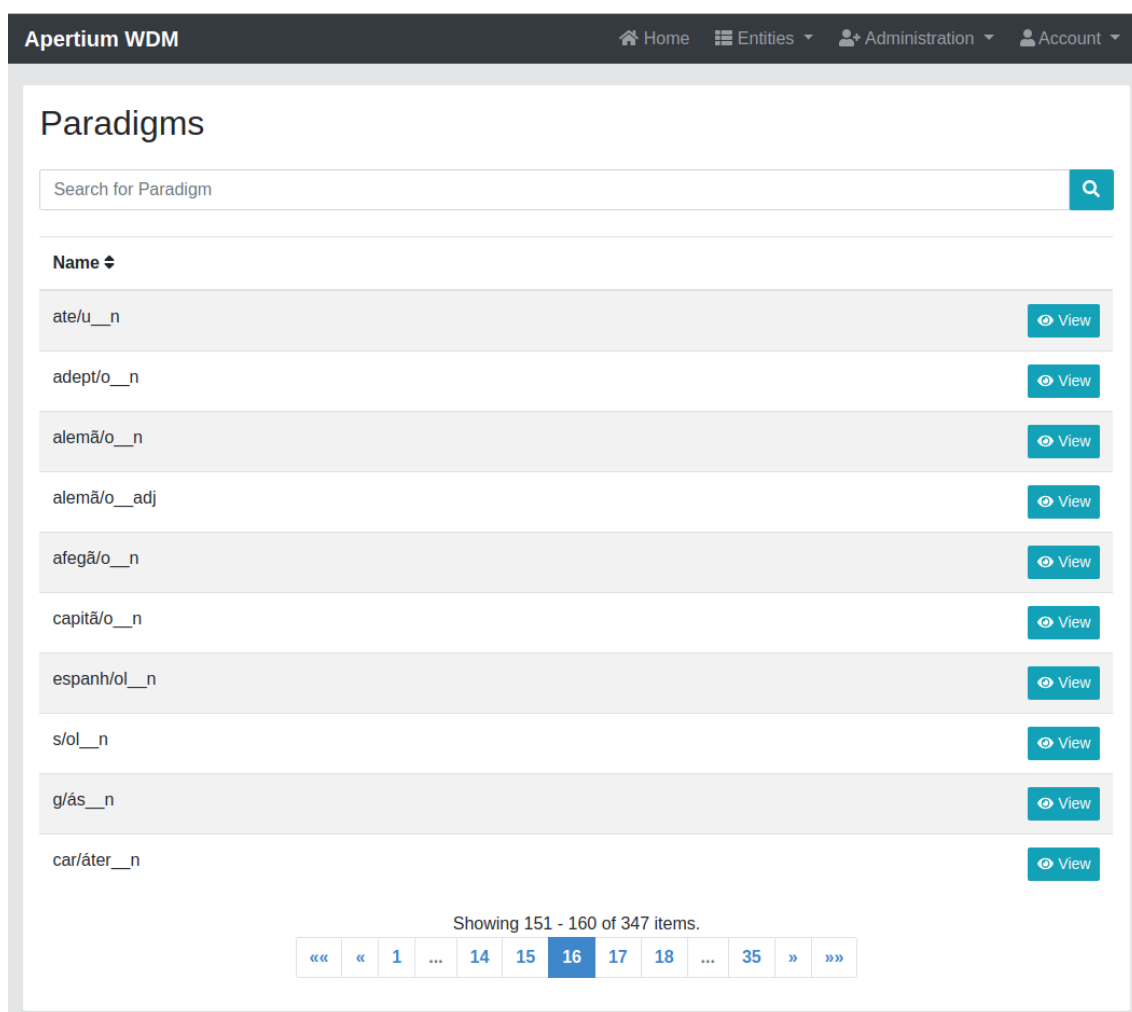
Search for Symbol

Name	Comment	
m	masculine	View
mf	masculine or feminine	View
sg	singular	View
pl	plural	View
adv	adverb	View
preadv	pre-adverb	View
pr	preposition	View
prn	pronoun	View
rel	relative	View
nn	inanimate	View

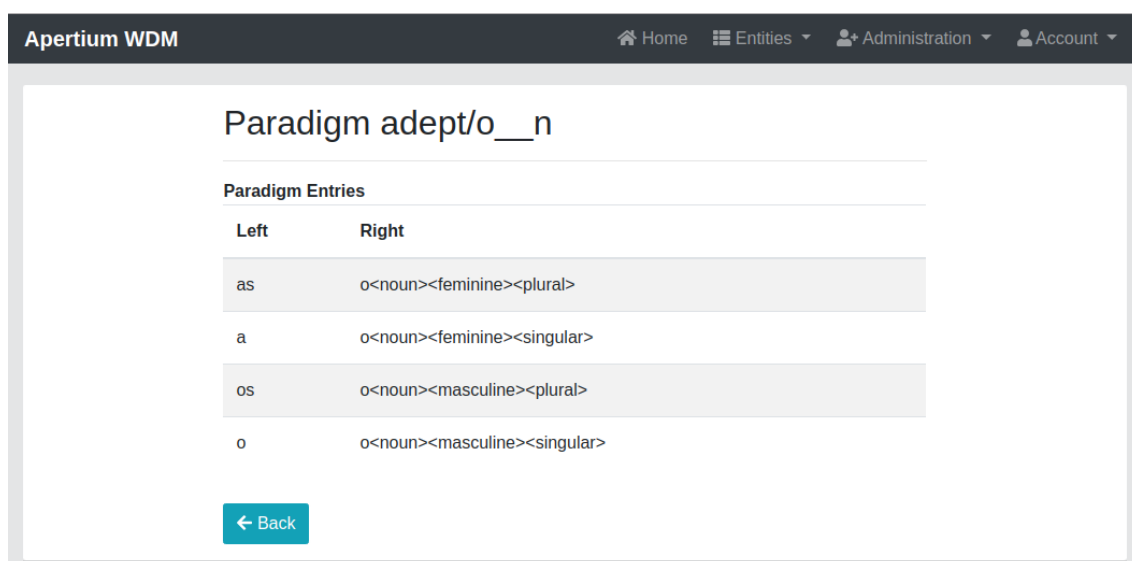
Showing 21 - 30 of 77 items.

[««](#)[«](#)[1](#)[2](#)[3](#)[4](#)[5](#)[...](#)[8](#)[»](#)[»»](#)

Tela de símbolos



Tela de paradigmas



Tela de detalhe de paradigma

Apertium WDM
Home
Entities
Administration
Account

Lemmas

[+ Create a new Lemma](#)

Name	Paradigm	Root	Line No	
outro	abert/o__adj		14654	View
aaleniano	abert/o__adj	aalenian	14655	View
ababalhável	aceitáve/l__adj	ababalháve	14656	View
ababosável	aceitáve/l__adj	ababosáve	14657	View
abacelável	aceitáve/l__adj	abaceláve	14658	View
abacial	aceitáve/l__adj	abacia	14659	View
abacinável	aceitáve/l__adj	abacináve	14660	View
abado	abert/o__adj	abad	14661	View
abadernável	aceitáve/l__adj	abadernáve	14662	View
abadessável	aceitáve/l__adj	abadessáve	14663	View

Showing 1 - 10 of 80707 items.

««
«
1
2
3
4
5
...
8071
»
»»

Tela de lemas