Radicalizador do português brasileiro baseado na Morfologia Distribuída

Brazilian Portuguese Stemmer Based on the Distributed Morphology

Gustavo Estivalet[1,2]
Karla Pinheiro[1,2]
José Ferrari-Neto[1,2]

RESUMO

A discussão teórica e empírica acerca do processamento morfológico tem sido relacionada às suas diferentes arquiteturas, enquanto os modelos de *Full Entry* propõem o acesso lexical por meio da representação da palavra inteira, os modelos de *Full Parsing* propõem o acesso lexical por meio da decomposição morfológica. Nesse sentido, radicalizadores (*stemmers*) são mecanismos úteis para a análise morfológica porque permitem a conflação de palavras e a modelagem computacional do processamento morfológico, assim como das interfaces morfofonológica e morfossintática. Contudo, os radicalizadores atuais de Processamento de Linguagem Natural foram desenvolvidos desconsiderando os modelos teóricos de Morfologia ou da Linguística. Sendo assim, o objetivo do presente trabalho foi desenvolver um radicalizador do português brasileiro baseado na teoria linguística da Morfologia Distribuída. Os objetivos específicos foram i) apresentar o repertório de morfemas flexionais e derivacionais, ii) analisar os morfemas em termos de traços morfossintáticos e classes gramaticais, e iii) modelar a decomposição de palavras polimorfêmicas. Nossos resultados apontam que um modelo decomposicional é capaz de processar as estruturas morfológicas do português brasileiro. Ainda, um modelo flexível com diferentes subestruturas para os casos de alomorfia poderia apresentar resultados mais eficazes para a derivação e a flexão irregular. Além da radicalização, nosso algoritmo apresenta algumas características específicas, tais como i) a representação dos traços morfossintáticos, ii) a definição da classe gramatical derivacional e iii) especificação teórica baseado na teoria. Enfim, o algoritmo possui código aberto e acesso livre para utilização em diversas aplicações, estando disponível em https://lexicodoportugues.shinyapps.io/Stemmer/.

Palavras-chave: *Radicalizador; Português brasileiro; Morfologia Distribuída*.

1 Universidade Federal da Paraíba (UFPB).
2 Laboratório de Processamento Linguístico (LAPROL)

ABSTRACT

The theoretical and empirical discussion about morphological processing has been related to its different architectures, while the Full Entry models propose lexical access through whole-word representation, the Full Parsing models propose lexical access through morphological decomposition. In this sense, stemmers are useful mechanisms for morphological analysis because they allow the confluence of words and the computational modeling of the morphological processing, as well as the morphophonological and morphosyntactic interfaces. However, the current stemmers from Natural Language Processing have been developed disregarding the theoretical models of Morphology or Linguistics. Therefore, the objective of the present work was to develop a Brazilian Portuguese stemmer based on the linguistic theory of Distributed Morphology. The specific objectives were i) present the repertoire of inflectional and derivational morphemes, ii) analyze the morphemes in terms of morphosyntactic features and grammatical classes, and iii) modeling the decomposition of polymorphemic words. Our results indicate that a decompositional model is capable of processing the morphological structures of Brazilian Portuguese. Still, a flexible model with different substructures for allomorphy cases could present more effective results for irregular derivation and inflection. In addition to stemming, our algorithm presents some specific characteristics, such as i) the representation of the morphosyntactic features, ii) the definition of the derivational grammatical class, and iii) a theoretical specification. The algorithm has open source and free access for use in several applications, being available at https://lexicodoportugues.shinyapps.io/Stemmer/.

Keywords: *Stemmer; Brazilian Portuguese; Distributed Morphology*.

1 Introdução

As línguas românicas originadas do latim vulgar, como o português, francês, espanhol, italiano e romeno, são línguas sintéticas que apresentam um rico sistema morfológico de derivação e flexão. Tal sistema é parte constituinte da competência linguística dos falantes que tem sido descrita em termos de uma gramática internalizada, cujo objetivo é fornecer um modelo teórico que explicite os elementos e as operações envolvidas na geração de representações linguísticas. Nesse sentido, diferentes tipos de modelos têm sido desenvolvidos com o objetivo de explicar o processamento morfológico e as representações linguísticas no léxico para o acesso lexical (CLAHSEN, 2006).

De uma forma geral, três tipos de modelos de acesso lexical têm sido propostos. Modelos de *Full Entry* (JACKENDOFF, 1975, p. 639, tradução nossa[3]) propõem que o indivíduo "desenvolva uma noção das regras lexicais de redundância lexical que tornam possível uma descrição adequada das

3 "[…] develop a notion of lexical redundancy rules which permits an adequate description of the partial relations and idiosyncrasy characteristic of the lexicon".

relações parcialmente idiossincráticas próprias do léxico". Em outras palavras, este modelo assume que a entrada lexical oferece previsibilidade, visto que se relaciona a outro item semelhante, assumindo entradas lexicais específicas e completas. Os modelos de *Full Parsing* (HALLE; MARANTZ, 1993, p. 111, tradução nossa[4]) propõem a "noção que afixos, assim como radicais lexicais, são peças mórficas cujas entradas lexicais relacionam forma fonológica com significado e função", ou seja, o acesso lexical ocorre através da decomposição morfológica. Enfim, os modelos de dupla-rota permitem as duas formas de processamento conforme parâmetros específicos da língua e das palavras. Nesse sentido, salienta-se que modelos de *Full Entry* pressupõem a necessidade de uma grande capacidade de armazenamento e poucas computações para o acesso lexical, diferentemente, modelos de *Full Parsing* pressupõem a necessidade de uma pequena capacidade de armazenamento e uma grande quantidade de computações para o acesso lexical (CLAHSEN, 2006).

Uma questão que surge daí é em que medida tais modelos podem ser replicados por meio de algoritmos computacionais que, por sua vez, poderiam ser utilizados para simular artificialmente as operações previstas nos respectivos modelos teóricos, como, por exemplo, o reconhecimento dos afixos e radicais presentes nas palavras. Algumas propostas nesse sentido já foram realizadas (VIERA; VIRGIL, 2007), contudo, uma característica comum a elas é que são desenvolvidas sob a perspectiva da eficácia computacional, muitas vezes ignorando características e propriedades linguísticas dos elementos e operações realizadas, o que faz com que se distanciem da replicação de mecanismos típicos da gramática de uma língua.

Sendo assim, o objetivo principal desta pesquisa foi programar um algoritmo computacional para o desenvolvimento de um radicalizador de palavras simples do português brasileiro (PB) baseado na teoria linguística da Morfologia Distribuída (MD) (HALLE; MARANTZ, 1993), permitindo a modelagem computacional da teoria, assim como a simulação do funcionamento do léxico através da decomposição morfológica mesmo em casos de idiossincrasia, alomorfismo, sincretismo, irregularidade e pseudopalavras. Portanto, os objetivos específicos foram i) apresentar o repertório de morfemas flexionais e derivacionais, ii) analisar os morfemas em termos de traços morfossintáticos e classes gramaticais, e iii) modelar a decomposição de palavras polimorfêmicas.

Destaca-se que, atualmente, os radicalizadores do PB existentes foram desenvolvidos para fins computacionais, tais como a criação de catálogos, mecanismos de busca, sumarização, sintetização e recuperação da informação. Dias e Melheiros (2005) propõem que o Processamento de Linguagem Natural (PLN) é a tecnologia computacional que busca estudar

4 "[…] notion that affixes as well as lexical stems are 'morpheme' pieces whose lexical entries relate phonological form with meaning and function".

e compreender a linguagem humana. Todavia, estes instrumentos têm sido desenvolvidos principalmente por pesquisadores e profissionais das áreas de engenharia computacional, engenharia da informação, análise/processamento de dados e comunicações (GOMES FERREIRA *et al.*, 2015; ORENGO; HUYCK, 2001; SOARES; PRATI; MONARD, 2009; VIERA; VIRGIL, 2007). Logo, estes radicalizadores desconsideram as perspectivas teóricas da Linguística e priorizam a eficácia empírica para fins específicos de forma sistêmica e superficial. Ou seja, eles estão interessados no produto final da radicalização, isto é, os radicais lexicais, mas não se interessam pelo processo de radicalização, especialmente através de uma perspectiva de análise, processamento, representação e decomposição morfológicos.

Conforme o nosso conhecimento, nenhum dos radicalizadores atuais do PB possui embasamento na análise linguística a partir de teorias e de modelos específicos de processamento morfológico. Portanto, este trabalho justifica-se por seu enfoque em considerar modelos teóricos e empíricos de funcionamento da linguagem como uma realidade psicológica (FERRARI-NETO, 2014), permitindo a modelagem e a simulação das teorias e modelos existentes, assim como se apresentando como um mecanismo computacionalmente justificado na teoria, sendo ainda mais eficaz na radicalização de palavras do que os demais algoritmos gerais.

Nesse sentido, a escolha do modelo da MD para o desenvolvimento do presente trabalho se deu pelos seguintes motivos: i) é um dos modelos decomposicionais pré-lexicais atuais em estudo e desenvolvimento, ii) é um dos poucos modelos morfológicos teóricos que apresenta respaldo na realidade psicolinguística e estudos empíricos, iii) é um modelo que apresenta detalhes sobre sua arquitetura e especificidades sobre seu funcionamento, permitindo a implementação de um algoritmo computacional para modelagem morfológica (EMBICK; HALLE, 2005). Além disso, esta ferramenta pode também ser explorada como instrumento didático para o estudo de questões linguísticas teóricas e empíricas relacionadas ao processamento morfológico e ao léxico sob a perspectiva da MD e da psicolinguística (FABRI *et al.*, 1995). Enfim, o presente algoritmo pretende contribuir de forma eficiente e confiável para a tarefa básica de radicalização de palavras no PLN e Linguística Computacional.

Os radicalizadores são mecanismos computacionais que decompõem as palavras, reconhecendo os afixos para apresentação do radical, base e/ou raiz (LOVINS, 1968). Seu intuito é automatizar o processo de conflação lexical, permitindo a combinação de palavras de uma língua em representações simples e gerais. Os radicalizadores são úteis em diversos domínios computacionais, "[...] pois permitem reduzir todas as palavras de um texto ou de uma pesquisa aos radicais que as compõem, de forma a agrupar por similaridade variações

Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

ortográficas que de outra forma passariam como palavras completamente distintas" (DIAS; MELHEIROS, 2005, p. 2).

2 Radicalizadores existentes para o PB

O conceito dos radicalizadores atuais foi estabelecido a partir do trabalho seminal de Lovins (1968) que, por sua vez, propôs um algoritmo simples e eficaz de radicalização das palavras do inglês. Em seguida, o algoritmo de Porter passou a ser largamente conhecido porque "embora simples, ele tem um desempenho ligeiramente melhor do que um sistema muito mais elaborado com o qual ele foi comparado. Ele funciona eficazmente tratando sufixos complexos como compostos feitos de sufixos simples e removendo o sufixo simples em várias etapas" (PORTER, 1980, p. 313, tradução nossa[5]). Portanto, o algoritmo de Porter não segue um modelo de decomposição morfológica propriamente dito, pois ele não funciona a partir de unidades morfológicas conforme os conceitos da Linguística e Morfologia, mas sim a partir de cadeias de caracteres pré-estabelecidas que representam combinações possíveis e frequentes das terminações das diferentes línguas.

Nessa perspectiva, Soares, Prati e Monard (2009) tinham como objetivo descrever a implementação do algoritmo de Porter para a radicalização de palavras do português, propondo melhorias no conjunto de regras através da inclusão de especificidades das características da língua portuguesa. Os resultados mostraram que o processamento do radicalizador proposto fora mais rápido que o algoritmo precedente sem ocasionar perdas na qualidade dos radicais gerados. Os autores ressaltam que as modificações mais relevantes estão relacionadas ao tratamento de verbos irregulares, pronomes oblíquos e ajustes das terminações derivacionais (SOARES; PRATI; MONARD, 2009). Entretanto, apesar do desempenho positivo mencionado, questões de ordem teórica e/ou linguística não são mencionadas, simplificando, portanto, a complexidade dos processos de formação de palavras e reduzindo o processamento morfológico para o acesso lexical ao apagamento de terminações discretas.

De forma semelhante, Orengo e Huyck (2001) exploraram o desenvolvimento de um algoritmo computacional de radicalização de palavras do português através do apagamento de terminações pré-estabelecidas. Diferentemente, esses autores implementaram o apagamento de diferentes tipos de afixos em diferentes etapas, assim como propuseram regras de ajuste para a normalização do radical. Os resultados mostraram que o radicalizador proposto comete menos erros nas derivações e nas palavras menores, sendo

5 "Although simple, it performs slightly better than a much more elaborate system with which it has been compared. It effectively works by treating complex suffixes as compounds made up of simple suffixes, and removing the simple suffix in a number of steps".

mais eficaz que a versão de base do algoritmo de Porter para o português (ORENGO; HUYCK, 2001). Contudo, destaca-se que as etapas propostas no algoritmo não consideram as diferentes terminações apagadas como unidades morfológicas, assim como as regras idiossincráticas de normalização do radical não consideram processos alomórficos produtivos.

Diferentemente, Alvares, Garcia e Ferraz (2005) tiveram como objetivo propor um algoritmo de radicalização de palavras do PB baseado no estudo estatístico das frequências das últimas letras das palavras. Conforme os autores, palavras são cadeias de letras organizadas de tal forma que podem representar significados, portanto radicalizadores estatísticos "não requerem nenhum conhecimento linguístico, sendo totalmente independente da estrutura morfológica da língua alvo" (ALVARES; GARCIA; FERRAZ, 2005, p. 696, tradução nossa[6]). Para tanto, eles implementaram um algoritmo que vai apagando sucessivamente as últimas letras de cadeias de caracteres em função das frequências das mesmas. Os resultados apontaram uma eficiência superior ao radicalizador de Orengo e Huyck (2001), contudo, fenômenos de sub- e sobrerradicalização foram observados, sendo dificilmente contornados sem a especificação de listas de palavras ou regras linguísticas.

Em seguida, o trabalho de Vieira e Virgil (2007) realizou uma revisão dos radicalizadores disponíveis para a língua portuguesa, sensibilizando os pesquisadores para a importância deste tipo de algoritmo para diversas finalidades da PLN e da Linguística Computacional. Conforme os autores, esses radicalizadores situam-se em um campo de pesquisa pouco explorado e com grande potencial, contudo, uma série de fatores limitantes e negativos desses algoritmos são apresentados, sugerindo reformulações consistentes em relação à arquitetura e implementação destes. Por exemplo, cada língua apresenta uma série de especificidades que devem ser consideradas no desenvolvimento do algoritmo, assim como sua extensão e seu funcionamento a novos processos de radicalização devem ser considerados. Enfim, os autores enfatizam que os radicalizadores atuais não são capazes de suprir as necessidades computacionais do português considerando plenamente seus processos morfológicos específicos e irregularidades lexicais (VIERA; VIRGIL, 2007).

O trabalho mais recente de Gomes Ferreira *et al.* (2015) explorou a metodologia de *hash-based* que por sua vez, apresenta uma tabela onde cada sufixo aponta para uma lista de regras. O algoritmo proposto apresentou resultados de eficiência semelhantes aos demais algoritmos, porém facilitando o processamento de palavras irregulares e a correção de erros através da programação de regras específicas associadas aos diferentes sufixos (GOMES FERREIRA *et al.*, 2015). Ainda assim, os autores não

6 "[…] does not require any linguistic knowledge whatsoever, being totally independent of the morphological structure of the target language".

Revista Letras, Curitiba, UFPR, n. 103, pp. 145-171, jan./jun. 2021
ISSN 2236-0999 (versão eletrônica)

apresentaram embasamento teórico para a determinação dos sufixos, regras de decomposição e alomorfia a partir de critérios e aspectos diretamente relacionados à estrutura morfológica das palavras.

Sendo assim, destaca-se a desconsideração e a não adequação desses algoritmos às teorias e modelos linguísticos e/ou morfológicos teóricos e/ou empíricos. Logo, fica clara a abordagem superficial geral a partir do apagamento de cadeias de caracteres, muitas vezes definidas de forma *ad hoc*, em metodologias de listagem e de frequência, não considerando aspectos específicos da formação de palavras do PB, assim como não implementando mecanismos recursivos para novas palavras e extensões do modelo.

Consequentemente, os algoritmos revisados acima não conseguem realizar a radicalização correta de palavras idiossincráticas, irregulares, alomórficas, pouco frequentes, neologismos, estrangeirismos, assim como de pseudopalavras que respeitem as regras grafotáticas do português (VIERA; VIRGIL, 2007). Mais do que isso, os algoritmos de radicalização atuais não apresentam nenhuma análise da decomposição morfológica, nem classificação dos morfemas e nem representação dos traços morfossintáticos, conforme previstos na maior parte das teorias linguísticas e morfológicas (VILLALVA, 2007). Enfim, nenhum dos radicalizadores analisados satisfaz os interesses da análise linguística em relação ao processamento de palavras polimorfêmicas, considerando a representação de sua estrutura hierárquica, os diferentes morfemas que compõem as palavras e os traços morfossintáticos ativados pelos morfemas funcionais das línguas naturais, não permitindo a modelagem e simulação da realidade psicológica do processamento morfológico e organização do léxico (FERRARI-NETO, 2014).

Logo, o radicalizador do português brasileiro (RadPB) proposto neste trabalho apresenta-se justamente como uma alternativa a esses algoritmos com o principal objetivo de preencher essas lacunas, permitindo o desenvolvimento de um algoritmo computacional baseado na teoria linguística da MD e proporcionando uma maior transparência entre o processo de radicalização e o resultado final do processamento morfológico de línguas naturais.

3 Aspectos importantes da Morfologia Distribuída

A principal característica da MD é que o léxico não é uma lista única contendo as representações das palavras a serem utilizadas nas sentenças; na MD, o léxico é distribuído em três listas. A Lista 1 contém os Traços Morfossintáticos, traços abstratos e raízes que não possuem materialização fonológica como morfemas lexicais ou afixos como morfemas. A Lista 2 contém os Itens do Vocabulário que, por sua vez, apresentam a representação fonológica dos morfemas associados aos traços da Lista 1. Enfim, a Lista 3 é

a Enciclopédia que contém o significado das palavras, assim como os demais conhecimentos não linguísticos (HALLE; MARANTZ, 1993).

Diferentemente de outros modelos de gramática gerativa, a MD rejeita a Hipótese Lexicalista no sentido que as palavras não são representadas como unidades inteiras com suas representações fonológicas, sintáticas e semânticas. Como consequência, a MD apresenta computações sintáticas que não operam em palavras extraídas do léxico, mas manipula traços morfossintáticos para gerar estruturas sintáticas (SIDDIQI, 2009). Desse modo, a derivação linguística atua em nós sintáticos através de operações sintáticas e operações morfológicas, sendo conhecida como *Syntactic Hierarchical Structure All the Way Down*.

Outro conceito importante da MD é a *Underspecification* dos Itens do Vocabulário que prevê que as entradas fonológicas não precisam ser completamente especificadas para a posição sintática (i.e., morfológica) em que são inseridas. Portanto, "não há necessidade de que as peças fonológicas de uma palavra forneçam os traços morfossintáticos dessa palavra; em vez disso, os Itens do Vocabulário são, em muitos casos, sinais *default*, inseridos onde nenhuma forma mais específica está disponível" (HARLEY; NOYER, 1999, p. 3, tradução nossa[7]). Em outras palavras, os Itens de Vocabulário da Lista 2 não precisam ter o feixe de traços morfossintáticos da Lista 1 completamente especificado para a inserção de material fonológico, permitindo, assim, a inserção de formas subespecificadas. Consequentemente, itens que têm mais especificação, ou seja, itens que são especificados para mais traços morfossintáticos, ganham a competição pela inserção fonológica.

Ainda, relacionado a este conceito, o caso *Elsewhere*, também conhecido como caso *default*, designa a aplicação de uma regra geral onde nenhuma outra regra ou especificação dos traços morfossintáticos foi satisfeita, obrigando o sistema a inserir um morfema *default*. Geralmente, esse caso é representado pelo morfema nulo, morfemas regulares, morfemas frequentes ou morfemas produtivos (SIDDIQI, 2009).

Enfim, um último conceito importante a ser explorado são as *Readjustment Rules* que ocorrem após a inserção fonológica dos Itens de Vocabulário da Lista 2. Assim, as regras de reajuste fonológico atuam sobre a forma final das palavras, permitindo adequações da forma para a pronúncia (i.e., *Spell-Out*) conforme o contexto linguístico. Nesse sentido, destaca-se que regras de alomorfia de palavras irregulares são aplicadas para o ajuste fonológico após a formação da palavra, assim como operações morfofonológicas simples são executadas apenas nesta etapa pós-lexical (HALLE; MARANTZ, 1994).

7 "[…] there is no need for the phonological pieces of a word to supply the morphosyntactic features of that word; rather, Vocabulary Items are in many instances default signals, inserted where no more specific form is available".

Do exposto acima, depreende-se que a MD se configura como um modelo teórico capaz de descrever não apenas as propriedades gramaticais dos morfemas envolvidos nas operações de estruturação das palavras de uma língua, mas também essas próprias operações. Portanto, sua adoção em algoritmos computacionais permitiria a modelagem lexical de uma forma que em muito se assemelharia ao que é observado na linguagem natural, trazendo confiabilidade, utilidade e eficiência a esse algoritmo. Portanto, esse é o intento que motivou o desenvolvimento da modelagem algorítmica exposta a seguir.

4. Desenvolvimento do RadPB

4.1 Algoritmo de Porter

Em relação à programação do algoritmo computacional do RadPB, partimos da estrutura básica do conhecido algoritmo de Porter que, por sua vez, segue sendo um dos algoritmos mais eficazes e versáteis para radicalização em diversas línguas (PORTER, 1980). Assim, são apresentados no Quadro 1 os conjuntos de terminações conforme o algoritmo de Porter para o PB (SOARES; PRATI; MONARD, 2009).

Quadro 1 – Algoritmo de Porter para o PB[8].

Processo	Apaga	Contextual	Ajuste
Etapa 1: Derivação	eza, ezas, ico, ica, icos, icas, ismo, ismos, ável, ível, ista, istas, oso, osa, osos, osas, amento, amentos, imento, imentos, adora, ador, ação, adoras, adores, ações, ante, antes, ância	amente, mente → iv, at, os, ic, ad _ idade, idades → abil, iv, iv _ iva, ivo, ivas, ivos → at _	logia, logias → log ução, uções → u ência, ências → ente ira, iras → ir
Etapa 2: Flexão	ada, ida, ia, aria, eria, iria, ará, ara, erá, era, irá, ava, asse, esse, isse, aste, este, iste, ei, arei, erei, irei, am, iam, ariam, eriam, iriam, aram, eram, iram, avam, em, arem, erem, irem, assem, essem, issem, ado, ido, ando, endo, indo, arão, erão, irão, ar, er, ir, as, adas, idas, ias, arias, erias, irias, arás, aras, erás, eras, irás, avas, es, ardes, erdes, irdes, ares, eres, ires, asses, esses, isses, astes, estes, istes, is, ais, eis, íeis, aríeis, eríeis, iríeis, áreis, areis, éreis, ereis, íreis, ireis, ásseis, ésseis, ísseis, áveis, ados, idos, ámos, amos, íamos, aríamos, eríamos, iríamos, áramos, éramos, íramos, ávamos, emos, aremos, eremos, iremos, ássemos, êssemos, íssemos, imos, armos, ermos, irmos, eu, iu, ou, ira, iras		
Etapa 3, 4, 5: Vogais	i → c _ os, a, i, o, á, í, ó e, é, ê		

8 https://snowballstem.org/algorithms/portuguese/stemmer.html.

Destaca-se que apenas a Etapa 1 do algoritmo de Porter possui processos de ajuste das terminações específicas substituídas por outras cadeias de caracteres, aproximando-se de processos de alomorfia do radical. Ainda, apenas a Etapa 1 e 3 possuem processos de apagamento baseados em características contextuais de precedência de cadeias de caracteres específicas às terminações indicadas. Assim, a regra geral de funcionamento do algoritmo é o apagamento das terminações listadas, com aplicação condicional da Etapa 1 ou da Etapa 2, ambas seguidas obrigatoriamente pela Etapa 3. Destaca-se que as terminações listadas no algoritmo de Porter são redundantes e não são equivalentes aos sufixos flexionais e derivacionais do PB, assim como o algoritmo não apresenta nenhum processo de decomposição de prefixos e infixos.

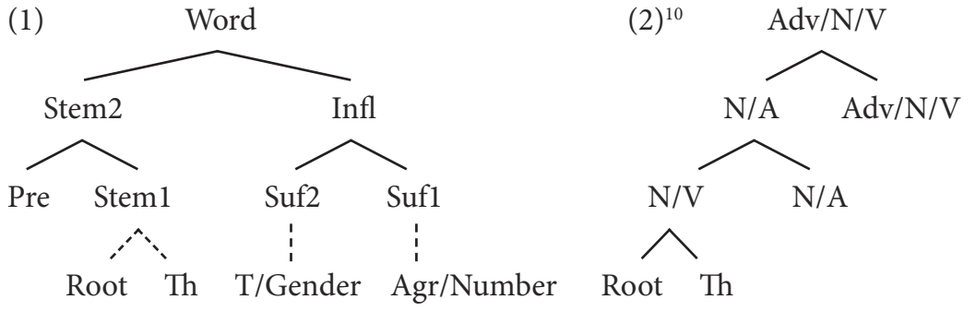
Por conseguinte, com o objetivo de realizarmos uma análise linguística a partir da MD, as terminações apresentadas conforme o algoritmo de Porter foram examinadas à luz da teoria linguística. Assim, as terminações contendo mais de um sufixo foram decompostas em função dos diferentes morfemas para a formação de palavras do PB, assim como outros sufixos e prefixos do PB (ROCHA, 1999) não presentes no algoritmo de Porter foram selecionados e analisados para complementação do algoritmo computacional do RadPB. Destaca-se que, diferentemente do proposto por Porter e da viabilidade de distribuírem-se os afixos entre flexionais e derivacionais para fins metodológicos, a MD desconsidera diferenças entre a flexão e a derivação (HARLEY; NOYER, 1999).

4.2 Análise morfológica para o RadPB

Para o desenvolvimento do RadPB, uma análise formal das estruturas hierárquicas morfológicas do PB foi realizada a partir da MD com o intuito de posteriormente explorarmos estas estruturas na radicalização das palavras assim como na aplicação de regras morfológicas para decomposição. Nesse sentido, três análises foram realizadas nas palavras do PB: i) flexão verbal, ii) flexão nominal e iii) derivação. Assim, através das análises linguísticas e definições das estruturas morfológicas, foi possível determinar o repertório de morfemas flexionais e derivacionais, sendo considerados os Itens do Vocabulário da Lista 2 da MD (HALLE; MARANTZ, 1993) e atingindo o nosso primeiro objetivo específico. Definimos como estrutura geral para palavras simples flexionadas do PB a estrutura conforme (1) e para palavras derivadas a estrutura conforme (2).

Destaca-se que a estrutura (2) também poderia ser utilizada para a flexão, pois, como resultado final, todos os morfemas das palavras são concatenados de forma linear, tendo as estruturas (1) e (2) apresentado o mesmo resultado. Contudo, a estrutura em (1) para a flexão parece representar

melhor os processos interativos, alomórficos e de percolação dos traços morfossintáticos entre os sufixos Suf1 e Suf2 para a flexão verbal e nominal[9] em um nó *Inflection* independente do radical (HALLE; MARANTZ, 1994).



Destaca-se que a partir dessas estruturas gerais, os morfemas podem ser hierarquicamente classificados durante o processo de decomposição e radicalização. Conforme (1), para a flexão verbal, devem-se considerar os morfemas de *Tense* e *Agrement*, já para flexão nominal, os morfemas de *Gender* e *Number*, para Suf2 e Suf1, respectivamente. Conforme (2), para a derivação devem-se considerar os sucessivos nós de derivação, onde em primeiro ciclo a *Root* se junta com a *Th* para a formação de um *Noun* ou *Verb*; no segundo ciclo, *Noun* ou *Verb* se junta com um morfema de *Noun* ou *Adjective*; e, no terceiro ciclo, *Noun* ou *Adjective* se junta com uma morfema de *Adverb*, *Noun* ou *Verb*, permitindo derivações com as transformações de classes gramaticais do PB (i.e., A → N/V/Adv, N → A/N/V/Adv, V → A/N).

4.3 Algoritmo do RadPB

Em seguida, com foco em nosso objetivo específico ii) analisar os morfemas em termos de traços morfossintáticos e classes gramaticais, ou seja, relacionar os Itens do Vocabulário da Lista 2 com os Traços Morfossintáticos da Lista 1 da MD, uma análise morfológica das características dos sufixos da flexão verbal (BASSANI; LUNGUINHO, 2011) e da flexão nominal foi realizada para a determinação dos traços morfossintáticos. Ainda, os sufixos derivacionais também foram analisados em função das classes gramaticais que cada sufixo cria em função da classe gramatical do radical precedente (EMBICK; HALLE, 2005). Além disso, foram considerados os principais

9 Neste trabalho, a flexão nominal engloba os processos de flexão de gênero e número de substantivos, adjetivos e pronomes.

10 Em (2), observa-se que os nós derivacionais N/A e Adv/N/V ainda poderiam ser divididos em elementos categorizadores Adv, A, N, V e o complemento Th, conforme Alcântara (2010). Contudo, preferimos manter a estrutura em (2) com o objetivo de preservarmos as representações dos morfemas categorizadores com as vogais temáticas conforme o Quadro 2, atendendo aos objetivos gerais e específicos dos radicalizadores.

prefixos e sufixos derivacionais produtivos a partir de estudos de morfologia teórica do português (BASILIO, 2011; ROCHA, 1999; VILLALVA, 2007). Assim, a partir do repertório de morfemas flexionais e derivacionais como Itens do Vocabulário da Lista 2 e do repertório de Traços Morfossintáticos da Lista 1, estabelecemos as relações entre ambas as listas, conforme o Quadro 2.

Quadro 2 – Repertório de morfemas e traços morfossintáticos do RadPB distribuídos nas Lista 1 – Traços Morfossintáticos e Lista 2 – Itens do Vocabulário, conforme a MD. Diferentes morfemas e traços estão separados por vírgula; traços *default* são apresentados em cinza[11].

Flexão			Derivação		
Nó	**Lista 2**	**Lista 1**	**Nó**	**Lista 2**	**Lista 1**
Agr	i	1, Singular	N|A	ão, zão, zarrão, ona, zona, inho, zinho	Grau
Agr	u	3, Singular	N|A	mente	Adv
Agr	m	3, Plural	N|A	dor, nte, eiro	N
Agr	mos	1, Plural	N|A	ção, gem, ismo, ância, idade, mento	N
Agr	s, ste	2, Singular	N|A	ar, er, ir, or, ôr, izar	V
Agr	is, stes, des	2, Plural	N|A	vel, âneo	A
T	va, ve, ia	Pretérito, perfeito	N|V	al, do, eza, ico, ivo, oso	N|A
T	ra	Pretérito, mais que perfeito	N|V	Ista	N|A
T	rá, re rão	Futuro, presente	N|V	logi, ência	N
T	ria, ría, ríe	Futuro, pretérito	Root	a, á, â, e, é, ê, i, í, o, ó, u, ú	Th
T	sse	Subjuntivo, imperfeito	Stem	a, im, in, re, pre, pré, sob, sub	Pre1
T	do	Particípio, passado	Stem	extra, inter, super, ultra	Pre2
T	ndo	Gerúndio			
T	r	Infinitivo			
Number	s, es	Plural			
Gender	a	Feminino			

No que diz respeito aos traços morfossintáticos especificados na Lista 1 da flexão, decidimos especificar redundantemente todos os traços presentes para cada morfema para fins de clareza e didáticos, portanto, os traços apresentados representam traços positivos presentes em cada um dos morfemas apresentados. Contudo, conforme o critério de *Underspecification* da MD, nem todos os traços precisam ser expressos para inserção do morfema na estrutura morfológica (HARLEY; BLANCO, 2013). Nesse sentido, nosso posicionamento é que um traço específico para cada nó sintático de (1) poderia ser considerado como *default*, resultando em uma diferença entre o número de traços representados para cada morfema e permitindo

11 Destaca-se que o Quadro 2 apresenta uma simplificação descritiva e teórica da representação dos nós sintáticos, morfemas e traços morfossintáticos visando aos objetivos específicos do presente trabalho. Para descrições teóricas mais completas, ver Bassani e Lunguinho (2011) e Embick e Halle (2005).

Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

a competição lexical para inserção dos Itens do Vocabulário da Lista 2 em função da especificidade do número de Traços Morfossintáticos da Lista 1. Sendo assim, nossa proposta é que os seguintes traços sejam considerados *default*, ou seja, sem marcação para os diferentes nós sintáticos: *Agreement* [3, Singular], *Tense* [Presente], *Aspect* [Perfeito], *Mood* [Indicativo], *Th* [C1], *Number* [Singular] e *Gender* [Masculino]. Esses traços estão representados em cinza no Quadro 2.

Ainda, observando-se o caso *Elsewhere* da MD, nosso posicionamento é que em relação à flexão o caso *default* deve ser representado pela inserção do morfema nulo nos nós morfológicos de *Agreement*, *Tense* e *Th* (e.g., ele fal-a-rá-$[ø]_{Agr}$, tu fal-a-$[ø]_{Tense}$-s, eu fal-$[ø]_{Th}$-ø-o,) (HARLEY; NOYER, 1999). No que diz respeito **à** derivação, o caso *default* seria a transformação de qualquer classe de palavra em *Noun*, com o morfema *default* para *Th* [o].

4.4 Funcionamento do algoritmo e da interface do RadPB

O algoritmo computacional do RadPB, as análises de *corpus* de palavras, a modelagem morfológica e as simulações do processamento morfológico foram realizados no programa R (ESTIVALET *et al.*, 2019). Com o objetivo de tornar o algoritmo acessível e permitir o uso do RadPB de forma simples e intuitiva, a biblioteca Shiny do R foi explorada para a criação de um aplicativo disponível na *Web*. Assim, para a utilização do RadPB, o usuário pode baixar o algoritmo computacional desenvolvido e utilizá-lo no seu computador no programa R ou simplesmente acessar o *link* disponibilizado e utilizar o RadPB através do aplicativo na *Web*.

Fica claro que além de um algoritmo trivial de radicalização, conforme os trabalhados revisados acima (VIERA; VIRGIL, 2007), o RadPB realiza também uma análise morfológica da palavra através da decomposição, das estruturas hierárquicas, dos morfemas e dos traços morfossintáticos. Tendo em vista a existência de palavras homógrafas e polissêmicas, as duas análises são realizadas e apresentadas: flexão nominal/verbal e derivação. No que diz respeito às palavras derivadas, o processamento da flexão nominal é realizado primeiramente, seguido do processamento da derivação (e.g., nacionalidades → nacionalidade$[s]_{Plural}$ → nacion$[al]_{Adj}[idade]_{Noun}$ (SIDDIQI, 2009).

O aplicativo da *Web* explora quatro funções básicas desenvolvidas no algoritmo computacional do RadPB. A função i) inflection.fnc decompõe os sufixos flexionais conforme a estrutura apresentada em (1) e de acordo com os morfemas apresentados na Lista 2 à esquerda do Quadro 2. A função ii) derivation.fnc decompõe os afixos derivacionais conforme a estrutura em (2) e de acordo com os morfemas apresentados na Lista 2 à direita do Quadro 2. A função iii) features.fnc utiliza as duas funções anteriores para ativar os traços morfossintáticos de cada morfema decomposto conforme as

relações entre a Lista 1 e a Lista 2 do Quadro 2. Enfim, a função iv) stemmer. fnc apaga todos os afixos à esquerda e à direita do radical, completando o processo de radicalização.

Sendo assim, o RadPB realiza a radicalização de palavras do PB através de construtos da MD, o algoritmo opera em ciclos, equivalentes às fases da derivação linguística (HARLEY; BLANCO, 2013). Na função i) inflection.fnc, primeiramente, o algoritmo procura à direita da palavra inteira (radical1) por morfemas flexionais possíveis de serem representados nos nós morfológicos *Agreement/Number*; caso encontre um morfema existente, o algoritmo decompõe este morfema do radical1, formando o radical2, e armazena o morfema decomposto em uma variável; caso o algoritmo não encontre nenhum morfema para este nó, o caso *Elsewhere* é aplicado e um morfema nulo é considerado. Em seguida, o algoritmo passa para o próximo ciclo a procura do morfema de *Tense/Gender* à direita do radical2, o mesmo processo é realizado sucessivamente até a decomposição da *Th* (FABRI *et al.*, 1995). Ao final, todos os morfemas armazenados nas variáveis conforme os respectivos ciclos são concatenados separados por *underline* "_" representando a decomposição morfológica.

Para a derivação, a função ii) derivation.fnc realiza o mesmo processo geral, contudo, a cada ciclo, os morfemas derivacionais formadores de palavras são procurados, decompostos e armazenados nas variáveis (i.e., *Noun*, *Adjective*, *Verb* e *Adverb*); ainda, o algoritmo realiza mais um ciclo à procura de Prefixos à esquerda do radical final (BASILIO, 2011). Em seguida, a partir da decomposição morfológica de ambas as funções descritas acima, a função iii) features.fnc ativa os respectivos traços morfossintáticos de cada morfema decomposto, conforme as relações estabelecidas no Quadro 2. Enfim, ainda sobre a estrutura decomposta, a função iv) stemmer.fnc apaga todos os afixos decompostos, completando o processo de radicalização. Portanto, o RadPB apresenta três informações de saída para cada palavra de entrada: i) radical, ii) decomposição morfológica, iii) traços morfossintáticos respectivos aos morfemas da palavra.

5 Desempenho e características do RadPB

Conforme explanado acima, o RadPB é muito mais do que um radicalizador. Radicalizadores têm como objetivo único apagar os sufixos flexionais, os sufixos derivacionais e, eventualmente, os prefixos e as vogais temáticas de palavras existentes para a determinação do radical da palavra (LOVINS, 1968). Em geral, estes mecanismos são desenvolvidos por cientistas da informação e de dados através da listagem de terminações, assim como de algoritmos computacionais que não representam a realidade linguística a partir de uma perspectiva teórica ou empírica do PLN (DIAS; MELHEIROS, 2005).

Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

Ultrapassando essas perspectivas, o RadPB realiza uma análise morfológica de qualquer palavra do PB, ou ainda de pseudopalavras que obedeçam à estrutura morfológica e grafotática do PB, a partir da teoria linguística da MD (HALLE; MARANTZ, 1994). Assim, o RadPB pode ser utilizado para a simulação da realidade psicológica do funcionamento da teoria, permitindo a modelagem, análise e uma melhor compreensão do sistema (FERRARI-NETO, 2014). Portanto, além de apresentar um desempenho otimizado como radicalizador, o RadPB tem seu valor pela própria forma como o algoritmo computacional foi programado com o objetivo de mimetizar um possível modelo de processamento morfológico de *Full Parsing* (HALLE; MARANTZ, 1993).

Observa-se que a maioria das abordagens utilizadas para a radicalização passa por soluções de listagem, frequentista ou *ad hoc*, conforme o próprio conhecido algoritmo de Porter (PORTER, 1980) que, de uma forma geral, apresenta um algoritmo muito simples com uma listagem redundante de terminações possíveis formadas por diversos e diferentes sufixos, conforme o Quadro 1. Mesmo considerando as recentes melhorias ao algoritmo de Porter para o PB (SOARES; PRATI; MONARD, 2009), os resultados continuam apresentando erros de radicalização sistemáticos que, por sua vez, dificilmente poderão ser resolvidos sem a utilização de especificidades da análise linguística.

Ainda nesse sentido, destaca-se que os outros radicalizadores atuais para o PB apresentaram abordagens frequentistas (ALVARES; GARCIA; FERRAZ, 2005), de listagem (ORENGO; HUYCK, 2001) ou de endereçamento (GOMES FERREIRA *et al.*, 2015), mas nenhum algoritmo de radicalizador explorou princípios linguísticos e parâmetros da língua portuguesa através de teorias e modelos linguísticos ou morfológicos, teóricos ou empíricos. Portanto, os resultados controversos e erros de radicalização destes mecanismos não aparecem como uma surpresa, mas como uma motivação para o desenvolvimento de radicalizadores que considerem a plausibilidade do PLN a partir da teoria linguística.

Em relação às abordagens de listagem largamente utilizadas para o desenvolvimento de radicalizadores (VIERA; VIRGIL, 2007), cabe-nos ressaltar que a possibilidade de modelos de *Full Entry* segue sendo pertinente para a representação no léxico a partir de perspectivas lexicalistas. Entretanto, esta abordagem, quando considerada dentro de uma teoria linguística, não analisa as palavras como simples cadeias de caracteres, conforme os radicalizadores atuais, mas as palavras devem ser consideradas através da sua representação lexical em pelo menos três níveis: fonológico, sintático e semântico (JACKENDOFF, 1975). Contudo, este modelo de representação lexical acaba desconsiderando um sistema gerativo de palavras com características dinâmicas e criativas de uso, além da invenção e da adaptação de palavras nas línguas naturais (VILLALVA, 2007).

Pois é justamente nesse sentido que o RadPB parece preencher as lacunas a que se propõe: desenvolver um algoritmo computacional de um radicalizador do PB baseado na MD. Assim, além de realizar a radicalização das palavras, ele realiza uma análise morfológica com a representação dos Traços Morfossintáticos dos afixos das palavras, portanto, uma parte significativa da informação sintática da Forma Lógica para a interpretação do sentido. Ainda, através das operações morfológicas para a decomposição (e formação de palavras), do repertório de morfemas como Itens do Vocabulário e das regras de reajuste morfofonológico, o RadPB provê outra parte significativa da informação fonológica (e ortográfica) para a interpretação do sentido das palavras. Enfim, ficam faltando as representações dos traços semânticos das raízes e o conhecimento não linguístico contido na Enciclopédia (Lista 3) da MD para a interpretação completa do sentido (SIDDIQI, 2009).

Assim, fica claro que o RadPB pode ser futuramente integrado a modelos gerais de processamento sintático, especialmente considerando as operações morfológicas dentro de uma perspectiva da *Syntax Hierarchical Srtructure All the Way Down* da MD. Nesse sentido, outros conceitos importantes à MD podem ser explorados, como a *Late Insertion* em que o conteúdo fonológico é inserido nos nós sintáticos apenas tardiamente durante a *Spell-Out*, que, por sua vez, é a operação em que os Itens do Vocabulário da Lista 2 são associados aos Traços Morfossintáticos da Lista 1. Relacionado a isso, a operação *Spell-Out* pode ser explorada sob a perspectiva da inserção e concatenação de morfemas lexicais (i.e., raízes e bases) e de morfemas funcionais (i.e., afixos) para pronúncia. Enfim, a operação de *Impoverishment* igualmente apresenta potencial de aplicação em radicalizadores através da perspectiva de enfraquecimento de traços morfossintáticos para seleção de afixos a serem decompostos, assim como as operações de *Fusion* e *Fission* também podem ser exploradas para a especificação de regras de alomorfia (HARLEY; NOYER, 1999).

O Quadro 3 apresenta exemplos de radicalização de palavras flexionadas e derivadas pelo RadPB, assim como as estruturas da decomposição morfológica e os traços morfossintáticos ativados pelos afixos.

Quadro 3 – Exemplos de palavras radicalizadas, decompostas e analisadas pelo RadPB. C1 – verbos da primeira classe, C3 – verbos da terceira classe, FP – futuro do presente do indicativo, PI – pretérito imperfeito do indicativo, 1P – primeira pessoa do plural, 3P – terceira pessoa do plural, FE – feminino, PL – plural, G – vogal temática, S – substantivo, A – adjetivo, Adv – advérbio.

Palavra	Radical	Decomposição	Traços
amaremos	am	am_a_re_mos	Root_C1_FP_1P
resurgiam	surg	re_surg_i_a_m	Pre_Root_C3_PI_3P
amigas	amig	amig_a_s	Root_FE_PL
reapagamento	apag	re_apag_a_mento	Pre_Root_G_S
nacionalmente	nacion	nacion_al_mente	Root_A_Adv

Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

6 Considerações finais

O presente trabalho apresentou o RadPB, um radicalizador do PB baseado na teoria linguística da MD (HALLE; MARANTZ, 1993). Nesse sentido, nosso objetivo foi programar um algoritmo computacional que simulasse os conceitos e funcionamento da teoria para a modelagem de dados empíricos do PB. Os resultados apontaram a possibilidade concreta de desenvolvimento de um radicalizador eficaz e simples a partir da teoria linguística tanto para a flexão (BASSANI; LUNGUINHO, 2011) como para a derivação de palavras polimorfêmicas.

Assim, nossos resultados e estudo apontam que as palavras do PB podem ser processadas e representadas conforme as premissas da MD, que, por sua vez, é um modelo de *Full Parsing* de via única com decomposição pré-lexical desenvolvido de acordo a arquitetura de Item e Processamento, baseada em estruturas morfológicas hierárquicas e regras de reajustes fonológicos (HARLEY; NOYER, 1999). Alternativamente, um modelo mais flexível que considere diferentes subestruturas dos radicais pode ser explorado para processar mais rapidamente e facilmente as supressões, idiossincrasias, irregularidades e processos alomórficos complexos no radical das palavras do PB (FABRI *et al.*, 1995).

Para a continuação deste trabalho, nosso foco será nos processos de alomorfia dos radicais irregulares. A proposta é minerar e analisar o funcionamento do RadPB no léxico do PB LexPorBR-Infantil (ESTIVALET *et al.*, 2019) com o objetivo de investigarem-se as consistências irregulares a partir de n-gramas de processos alomórficos do radical, comparando os resultados encontrados com outros radicalizadores existentes para o PB.

Enfim, esperamos que o RadPB possa se desenvolver ainda mais a partir de outras premissas teóricas da MD, que apresente resultados mais eficazes que os atuais radicalizadores do PB e que possa ser explorado como um instrumento didático para a pesquisa e o estudo do processamento morfológico. O algoritmo computacional e o aplicativo desenvolvido em linguagem R e Shinny possuem código aberto e acesso livre e estão disponíveis através de uma interface na *Web* https://lexicodoportugues.shinyapps.io/Stemmer/.

Agradecimentos

Agradecemos a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa do segundo autor. Agradecemos à Comissão Organizadora do V Colóquio Brasileiro de Morfologia pela organização e realização do evento. Enfim, agradecemos enormemente a dois revisores

anônimos da Revista Letras da UFPR pela leitura atenciosa do artigo e suas considerações pertinentes.

Referências

ALCÂNTARA, C. C. As classes formais do português brasileiro. *Letras de Hoje*, v. 45, n. 1, p. 6–15, 2010.

ALVARES, R. V.; GARCIA, A. C. B.; FERRAZ, I. STEMBR: A stemming algorithm for the Brazilian Portuguese language. *Lecture Notes in Computer Science*, v. 3808, p. 693–701, 2005.

BASILIO, M. *Formação e classes de palavras no português do Brasil*. São Paulo, Brasil: Editora Contexto, 2011.

BASSANI, I. S.; LUNGUINHO, M. V. Revisitando a flexão verbal do português à luz da Morfologia Distribuída: um estudo do presente, pretérito imperfeito e pretérito perfeito do indicativo. *Revista Virtual de Estudos da Linguagem*, v. ed. esp., n. 5, p. 199–227, 2011.

CLAHSEN, H. Dual-Mechanism Morphology. In: BROWN, K. (Ed.). *Encyclopedia of Language & Linguistics*. Oxford: Elsevier, p. 1–5, 2006.

DIAS, M. A. L.; MELHEIROS, M. G. *Estudo de técnicas de radicalização para Língua Portuguesa*, 2005.

EMBICK, D.; HALLE, M. *On the status of stems in morphological theory.* In: GEERTS, T.; JACOBS, H. (Eds.). Procedings of Going Romance 2003. Amsterdam: John Benjamins, 2005.

ESTIVALET, G. L.; HARTMANN, N. S.; MARQUIAFAVEL, V.; LUKASOVA, K.; CARTHERY-GOULART, M. T.; ALUÍSIO, S. M. *LexPorBR Infantil: Uma base lexical tripartida e com interface Web de textos ouvidos, produzidos e lidos por crianças*. In: PROLO, C.; OLIVEIRA, L. (Eds.). Proceedings of the XII Symposium in Information and Human Language Technology (STIL2019). Salvador-BA, p. 190-199, 2019.

FABRI, R.; RUMP, C.; URBAS, M.; WALTHER, M. A computational Model of Minimalist Morphology. *Theorie des Lexikons*, v. 62, p. 1–72, 1995.

FERRARI-NETO, J. Explorando as relações entre léxico mental e gramática: processamento morfológico num enfoque integrado. In: ARAGÃO NETO, M.; CAMBRUS, M. (Eds.). *Léxico e gramática: novos estudos de interface*. Curitiba-PR: CRV, p. 13–41, 2014.

GOMES FERREIRA, W. SANTOS, W. A.; SOUZA, B. M. P.; ZAIDAN, T. M. M.; BRANDÃO, W. C. Assessing the Efficiency of Suffix Stripping Approaches for Portuguese Stemming. In: ILIOPOULOS, C.; PUGLISI, S.; YILMAZ, E. (Eds.). *Lecture Notes in Computer Science*. Springer International Publishing, v. 9309, p. 210–221, 2015.

HALLE, M.; MARANTZ, A. Distributed morphology and the pieces of inflection. In: HALE, K.; KEYSER, S. J. (Eds.). *The view from building 20: essays in linguistics in honor of Sylvain Bromberger*. Cambridge, Massachusetts: The MIT Press, p. 111–176, 1993.

HALLE, M.; MARANTZ, A. Some key features of distributed morphology. *MIT Working Papers in Linguistics*, v. 21, n. Papers on Phonology and Morphology, p. 275–288, 1994.

HARLEY, H.; BLANCO, M. T. Cycles, Vocabulary Items and Stem Forms in Hiaki. In: MATUSHANSKY, O.; MARANTZ, A. (Eds.). *Distributed Morphology Today*. Cambridge-MA: The MIT Press, p. 117–132, 2013.

HARLEY, H.; NOYER, R. State-of-the-Article: Distributed Morphology. *Glot International*, v. 4, n. 4, p. 3–9, 1999.

JACKENDOFF, R. Morphological and Semantic Regularities in the Lexicon. *Language*, v. 51, n. 3, p. 639-671, 1975.

LOVINS, J. B. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, v. 11, p. 22-31, 1968.

ORENGO, V. M.; HUYCK, C. *A stemming algorithm for the portuguese language*. Proceedings Eighth Symposium on String Processing and Information Retrieval. IEEE, p. 186-193, 2001

PORTER, M. F. *An algorithm for suffix strippingProgram*, p. 313-316, 1980.

ROCHA, L. C. DE A. *Estruturas Morfológicas do Português*. Belo Horizonte-MG: Editora UFMG, 1999.

SIDDIQI, D. *Syntax within the Word: Economy, allomorphy, and argument selection in distributed morphology*. Amsterdam: John Benjamins Publishing Company, v. 138, 2009.

SOARES, M. V. B.; PRATI, R. C.; MONARD, M. C. Improvement on the Porter's Stemming Algorithm for Portuguese. *IEEE Latin America Transactions*, v. 7, n. 4, p. 472–477, 2009.

VIERA, A. F. G.; VIRGIL, J. Uma revisão dos algoritmos de radicalização em língua portuguesa. *Information Research*, v. 12, n. 3, p. 1–12, 2007.

VILLALVA, A. *Morfologia do Português*. Lisboa: Universidade Aberta, 2007.

Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

Anexo

Radicalizador do português brasileiro baseado na Morfologia Distribuída Brazilian Portuguese Stemmer Based on the Distributed Morphology

```
#################################################################
############
# Brazilian Portuguese Stemmer                                  #
#################################################################
############
# Clean all
rm(list=ls())

# Define a word
word <- "exemplos"

# Decomposer: verbal inflection
vinflection.fnc <- function(word){
  # Agreement
  stem1 <- sub("stes$|des$|mos$|ste$|is$|i$|m$|s$|u$", "", word)
  agr <- substr(word, nchar(stem1) + 1, nchar(word))
  # Tense
  stem2 <- sub("ndo$|rão$|ria$|ría$|ríe$|sse$|do$|ia$|ra$|rá$|re$|va$|ve$|r$",
"", stem1)
  tense <- substr(stem1, nchar(stem2) + 1, nchar(stem1))
  # Th
  stem3 <- sub("a$|á$|e$|ê$|i$|í$|o$|ô$", "", stem2)
  th <- substr(stem2, nchar(stem3) + 1, nchar(stem2))
  # Recompose
  morpho <- paste(stem3, th, tense, agr, sep = "_")
  morpho <- gsub("_+", "_", morpho)
  morpho <- gsub("_$", "", morpho)
  return(morpho)
}
vinflection.fnc(word)

# Features: verbal inflection
vfeat.fnc <- function(word){
  feat <- decomp.fnc(word)
  feat <- paste0("_", feat, "_")
```

```
 # Agreement node
 feat <- sub("_stes_", "_2P_", feat)
 feat <- sub("_des_", "_2P_", feat)
 feat <- sub("_mos_", "_1P_", feat)
 feat <- sub("_ste_", "_2S_", feat)
 feat <- sub("_is_", "_2P_", feat)
 feat <- sub("_i_", "_1S_", feat)
 feat <- sub("_m_", "_3P_", feat)
 feat <- sub("_s_", "_2S_", feat)
 feat <- sub("_u_", "_3S_", feat)
 # Tense node
 feat <- sub("_ndo_", "_GER_", feat)
 feat <- sub("_rão_", "_SF_", feat)
 feat <- sub("_ria_", "_PF_", feat)
 feat <- sub("_ría_", "_PF_", feat)
 feat <- sub("_ríe_", "_PF_", feat)
 feat <- sub("_sse_", "_SI_", feat)
 feat <- sub("_do_", "_PP_", feat)
 feat <- sub("_ia_", "_IP_", feat)
 feat <- sub("_ra_", "_PQ_", feat)
 feat <- sub("_rá_", "_SF_", feat)
 feat <- sub("_re_", "_SF_", feat)
 feat <- sub("_va_", "_IP_", feat)
 feat <- sub("_ve_", "_IP_", feat)
 feat <- sub("_r_", "_IF_", feat)
 # Th node
 feat <- sub("_a_|_á_", "_C1_", feat)
 feat <- sub("_e_|_ê_", "_C2_", feat)
 feat <- sub("_i_|_í_", "_C3_", feat)
 feat <- sub("_o_|_ô_", "_C4_", feat)
 # Finish
 feat <- sub("^_[[:alpha:]]*_", "_Root_", feat)
 feat <- gsub("_+", "_", feat)
 feat <- gsub("^_|_$", "", feat)
 return(feat)
}
vfeat.fnc(word)

# Decomposer: nominal/adjectival inflection
ninflection.fnc <- function(word){
 # Plural
 stem1 <- sub("([aéóu])is$", "\\1l", word) #[aéóu]is -> [aeou]l
```


Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

```
  stem1 <- sub("él$", "el", stem1)
  stem1 <- sub("ól$", "ol", stem1)
  stem1 <- sub("([nrz])es$", "\\1", stem1) #[nrz]es -> [nrz]
  stem1 <- sub("ães$|ões$", "ão", stem1) #ões|ães -> ão
  stem1 <- sub("ns$", "m", stem1) #ns -> m
  stem1 <- sub("s$", "", stem1)
  number <- substr(word, nchar(stem1) + 1, nchar(word))
  # Gender
  stem2 <- sub("o$|a$", "", stem1)
  gender <- substr(stem1, nchar(stem2) + 1, nchar(stem1))
  # Recompose
  morpho <- paste(stem2, gender, number, sep = "_")
  morpho <- gsub("_+", "_", morpho)
  morpho <- gsub("_$", "", morpho)
  return(morpho)
}
ninflection.fnc(word)

# Features: nominal/adjectival inflection
nfeat.fnc <- function(word){
  stem <- stemmer.fnc(word)
  feat <- substr(word, nchar(stem) + 1, nchar(word))
  # Number
  feat <- sub("_os_", "_MA_PL_", feat)
  feat <- sub("_as_", "_FE_PL_", feat)
  feat <- sub("_s_|_es_", "_PL_", feat)
  # Gender
  feat <- sub("_o_", "_MA_SG_", feat)
  feat <- sub("_a_", "_FE_SG_", feat)
  feat <- ifelse(nchar(feat) == 0, "_SG_", feat)
  # Finish
  feat <- sub("^_[[:alpha:]]*_", "_Root_", feat)
  feat <- gsub("_+", "_", feat)
  feat <- gsub("^_|_$", "", feat)
  return(feat)
}
nfeat.fnc(word)

# Decomposer: derivation
derivation.fnc <- function(word){
  # Degree
```

```
  stem1 <- sub("zarrão$|zarrões$|z?inh[oa]s?$|z?onas?$|z?ões$|z?ão$", "",
word)
 degree <- substr(word, nchar(stem1) + 1, nchar(word))
 # Adverb
 stem2 <- sub("mente$", "", stem1)
 adv <- substr(stem1, nchar(stem2) + 1, nchar(stem1))
 # Noun 2
 stem3 <- sub("âncias?$", "", stem2)
 stem3 <- sub("idades?$", "", stem3)
 stem3 <- sub("mentos?$", "", stem3)
 stem3 <- sub("eir[oa]s?$", "", stem3)
 stem3 <- sub("ismos?$", "", stem3)
 stem3 <- sub("idão$|idões$", "", stem3)
 stem3 <- sub("dor$|dores$|doras?$", "", stem3)
 stem3 <- sub("ção$|ções$", "", stem3)
 stem3 <- sub("gem$|gens$", "", stem3)
 stem3 <- sub("ntes?$", "", stem3)
 noun2 <- substr(stem2, nchar(stem3) + 1, nchar(stem2))
 # Adjective 2
 stem4 <- sub("âneos?$", "", stem3)
 stem4 <- sub("istas?$", "", stem4)
 stem4 <- sub("vel$|veis$", "", stem4)
 adj2 <- substr(stem3, nchar(stem4) + 1, nchar(stem3))
 # Verb
 stem5 <- sub("iz[aá]r?$", "", stem4)
 stem5 <- sub("[aeioô]r$", "", stem5)
 verb <- substr(stem4, nchar(stem5) + 1, nchar(stem4))
 # Adjective 1
 stem6 <- sub("ic[oa]s?$", "", stem5)
 stem6 <- sub("iv[oa]s?$", "", stem6)
 stem6 <- sub("os[oa]s?$", "", stem6)
 stem6 <- sub("ezas?$", "", stem6)
 stem6 <- sub("d[oa]s?$", "", stem6)
 stem6 <- sub("al$|ais$", "", stem6)
 adj1 <- substr(stem5, nchar(stem6) + 1, nchar(stem5))
 # Noun 1
 stem7 <- sub("logias?$", "log", stem6)
 stem7 <- sub("ência?$", "ente", stem7)
 noun1 <- substr(stem6, nchar(stem7) + 1, nchar(stem6))
 # Th
 stem8 <- sub("[aáâeéêiíoóuú]$", "", stem7)
 stem8 <- sub("ç$", "c", stem8)
```


Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)

```
  th <- substr(stem7, nchar(stem8) + 1, nchar(stem7))
  # Prefixes
  stem9 <- sub("^extra|^inter^|^super|^ultra", "", stem8)
  pre <- substr(stem8, 1, nchar(stem9) - nchar(stem8))
  # Recompose
  morpho <- paste(pre, stem9, th, noun1, adj1, verb, adj2, noun2, adv, degree,
sep = "_")
  morpho <- gsub("_+", "_", morpho)
  morpho <- gsub("_$", "", morpho)
  return(morpho)
}
derivation.fnc(word)

# Features: decompositional/morphological parser
dfeat.fnc <- function(word){
  feat <- decomp.fnc(word)
  feat <- paste0("_", feat, "_")
  # Degree
   feat <- sub("_zarrão_|_zarrões_|_z?inh[oa]s?_|_z?onas?_|_z?ões_|_z?ão_",
"_Deg_", feat)
  # Adverb
  feat <- sub("_mente_", "_Adv_", feat)
  # Noun
    feat <- sub("_log_|_ência?_|_âncias?_|_idades?_|_mentos?_|_ismos?_|_
dor_|_dores_|_doras?_|_ção_|_ções_|_gem_|_gens_|_ntes?_", "_Noun_",
feat)
  # Verb
  feat <- sub("_izar_|_[aeio]r_", "_Verb_", feat)
  # Adjectif
    feat <- sub("_istas?_|_vel_|_veis_|_ic[oae]s?_|_iv[oa]s?_|_os[oa]s?_|_
ezas?_|_d[oa]s?_|_al_", "_Adj_", feat)
  # Th
  feat <- sub("_[aáâeéêiíoóuú]_", "_Th_", feat)
  # Finish
  feat <- sub("^_[[:alpha:]]*_", "_Root_", feat)
  feat <- gsub("_+", "_", feat)
  feat <- gsub("^_|_$", "", feat)
  return(feat)
}
dfeat.fnc(word)

# Stemmer
```

```
stemmer.fnc <- function(word){
  stem <- decomp.fnc(word)
  stem <- sub("_.*", "", stem)
  return(stem)
}
stemmer.fnc(word)
```


Revista Letras,
Curitiba, UFPR,
n. 103, pp. 145-171,
jan./jun. 2021
ISSN 2236-0999
(versão eletrônica)