Parser sintático para o português brasileiro: desafios e soluções

Syntactic parser for Brazilian Portuguese: challenges and solutions

Willian Emerson Afonso Pacheco *[1] **e Manoel Francisco Guaranha** †[1,2]

[1] *Faculdade de Tecnologia do Estado de São Paulo, Câmara de Ensino, Extensão e Pesquisa da FATEC Ipiranga, São Paulo, SP, Brasil.*
[2] *Universidade Santo Amaro, Programa de Mestrado em Ciências Humanas, São Paulo, SP, Brasil.*

Resumo

Este artigo tem como objetivo apresentar o Parser Sintático para o Português Brasileiro – *Parsero*, desenvolvido a partir da Gramática Gerativa (CHOMSKY, 2015), aperfeiçoada pela Teoria X-Barra (CHOMSKY, 2014). Para tanto, foram utilizadas as regras desenvolvidas especialmente para o Português Brasileiro por Othero (2009) e adaptadas pelo nosso projeto para atender às necessidades de nosso Parser. A pesquisa utilizou como coleção lexical, para povoar um Banco de Dados *Structured Query Language* (SQL), o recurso *Dicionário de Palavras Simples Flexionadas para o Português Brasileiro* (DELAF_PB), disponibilizado pelo Projeto Unitex-PB, desenvolvido pelo Núcleo Interinstitucional de Linguística Computacional (NILC) e pelo Instituto de Ciências Matemáticas e de Computação (ICMC). Esse recurso, por sua vez, foi construído com base no formalismo francês – *Dictionnarie Electronique du LADL* (DELA) (MUNIZ, 2004). Como resultado, disponibilizamos a Base de Dados SQL com 1.193.295 unidades léxicas classificadas, o endereço com o código aberto do *Parsero* e um *link* para execução do aplicativo. Para desenvolver o Processador de Linguagem Natural (PLN), colocamos em prática estudos interdisciplinares em ciências da linguagem e ciências da computação, práticas necessárias para o desenvolvimento de programas inteligentes que consigam interagir com escritores e falantes do Português Brasileiro.

Palavras-chave: Linguística computacional. Processamento de Linguagem Natural. Gramática gerativa. Parser sintático. Português brasileiro.

Abstract

This article aims to present the Syntactic Parser for Brazilian Portuguese – Parsero –, developed from the Generative Grammar (CHOMSKY, 2015) improved by the X-Barra Theory (CHOMSKY, 2014). Therefore, the rules developed by Othero (2009) especially for Brazilian Portuguese were used and adapted by our project to meet the needs of our Parser. The research used as lexical collection, to populate a *Structured Query Language* (SQL) Database, the resource *Dictionary of Simple Inflected Words for Brazilian Portuguese* (DELAF_PB), which was made available available by the Unitex-PB Project, developed by Núcleo Interinstitucional de Linguística Computacional (NILC) and by Instituto de Ciências Matemáticas e de Computação (ICMC). This resource, in turn, was built based on the French formalism – *Dictionnarie Electronique du LADL* (DELA) (MUNIZ, 2004). As a result of our project, we have made available to researchers interested in the topic the SQL Database with 1,193,295 classified lexical units, the address with the open source of *Parsero* and a link to run the application. Throughout the development of the Natural Language Processor (NLP), we had to put into practice interdisciplinary studies from language sciences and computer sciences, a necessary practice for the development of intelligent programs that can interact with writers or Brazilian Portuguese speakers.

Keywords: Computational linguistics. Natural Language Processing. Generative Grammar. Syntactic parser. Brazilian Portuguese.

Textoline
Linguagem e Tecnologia

DOI: 10.35699/1983-3652.2022.37569

Seção:
Artigos

Autor Correspondente:
Manoel Francisco Guaranha

Editor de seção:
Danniervelin Pereira
Editor de layout:
Leonado Araújo

Recebido em:
20 de dezembro de 2021
Aceito em:
17 de março de 2022
Publicado em:
14 de maio de 2022

1 Considerações iniciais

Um dos problemas mais instigantes na área de processamento computacional é o de construção de *Parsers* sintáticos, sistemas que, dada uma sentença ou mais, produzem suas árvores sintáticas. Os

*Email: willian.pacheco@fatec.sp.gov.br
†Email: manoel.guaranha@gmail.com

desafios que se colocam na construção desses leitores eletrônicos compreendem as seguintes etapas: a construção e manutenção de um conjunto de itens lexicais organizados de modo a facilitar o processamento, já que o conjunto de palavras de uma língua é extenso e, pode-se dizer, aberto a novos termos que surgem a partir dos diferentes usos; a construção de algoritmos morfossintáticos, capazes de rotular os itens lexicais segundo sua função, uma vez que o valor posicional de cada unidade ou locução varia de acordo com a posição que ocupa nos enunciados; a construção de algoritmos capazes de agrupar os itens lexicais em sintagmas, atribuindo-lhes as funções sintáticas e validando-os segundo as regras de combinação estabelecidas pela gramática. Em cursos tecnológicos de Análise e Desenvolvimento de Sistemas, a investigação de processos de construção de *Parsers* sintáticos constitui uma tarefa interdisciplinar que oferece oportunidade de aprendizagem aos pesquisadores, tanto aos que se dedicam ao estudo das estruturas da língua, português falado no Brasil, no nosso caso, quanto àqueles que investigam estruturas de dados e de linguagens de programação que sejam adequadas para o Processamento de Linguagem Natural (PLN) que, diferentemente do processamento de números, sujeito às regras rígidas da Matemática, requerem processos mais complexos de tratamento. Sendo assim, o projeto descrito neste artigo disponibilizou, para pesquisadores interessados no tema, o aplicativo de PLN *Parsero*, e o Banco de Dados com o léxico do Português Brasileiro.

Este artigo pretende apresentar, portanto, o processo de elaboração do Parser Sintático para o Português Brasileiro - *Parsero*, que analisa sentenças escritas, bem como seus resultados, obtidos por meio de uma pesquisa realizada no Curso Superior de *Tecnologia em Análise e Desenvolvimento de Sistemas* da Faculdade de Tecnologia do Estado de São Paulo (FATEC), no âmbito da *Câmara de Ensino, Pesquisa e Extensão* CEPE/Fatec Ipiranga). Para tanto, foram utilizados os conceitos da Gramática Gerativa de Chomsky (2014, 2015) (originais publicados em 1957 e 2012, respectivamente), com adaptações propostas por Othero (2009) para atender às especificidades do Português Brasileiro e, além disso, com adaptações que foram necessárias para atender às especificidades deste projeto. A pesquisa utilizou como recurso para construção de um banco de dados do léxico o *Dicionário de Palavras Simples Flexionadas para o Português Brasileiro* (DELAF_PB), arquivo com aproximadamente 880.000 palavras. Esse recurso, cuja documentação detalhada pode ser encontrada em Muniz (2004), foi disponibilizado pelo Projeto Unitex-PB, desenvolvido pelo *Núcleo Interinstitucional de Linguística Computacional* (NILC) e pelo *Instituto de Ciências Matemáticas e de Computação* (ICMC) e foi construído com base no formalismo francês *Dictionnarie Electronique du LADL* (DELA) (MUNIZ, 2004).

A pesquisa desenvolveu, documentou e disponibilizou em repositórios virtuais três recursos linguístico-computacionais para o Português Brasileiro: o Banco de Dados SQL (*Structured Query Language*) ParseroDB (PACHECO; GUARANHA, 2021a,b); o código aberto do processador sintático *Parsero* (PACHECO; GUARANHA, 2021b); e uma interface visual para execução do *software* (PARSERO:..., 2021). Espera-se que esses recursos possam ser utilizados e, eventualmente, aperfeiçoados por pesquisadores que se interessem em avançar nos trabalhos de desenvolvimento de sistemas na área de Processamento de Linguagem Natural para o Português Brasileiro.

Para dar conta de nosso objetivo de apresentar o processo de construção do *Parsero* e seus resultados, este artigo será desenvolvido em três seções, além das considerações iniciais e finais, numeradas como seção 1 e seção 5. Na seção 2, discutiremos aspectos gerais da Gramática Gerativa proposta por Chomsky (2015) e aperfeiçoada pela Teoria X-Barra, conforme afirma o próprio Chomsky (2014, p. 392). Na seção 3, apresentaremos o Banco de Dados SQL que serviu de corpus para o nosso Processador de Linguagem Natural, detalhando o processo de construção desse recurso. Na seção 4, apresentaremos as características de nosso *Parser*, o *Parsero*, e as regras da Gramática Gerativa adaptadas para o Português Brasileiro por Othero (2009)[1], que foram utilizadas no algoritmo, justificando, ainda, algumas adaptações que fizemos para atender às nossas necessidades. Nessa última seção também apresentaremos exemplos de sentenças processadas pelo aplicativo.

1 Optamos por utilizar em nosso projeto os termos em português "Sintagma Verbal" (SV), "Sintagma Nominal" (SN), e assim por diante, em lugar de "*Verb Phrase*" (VP), "*Noun Phrase*" (NP) etc., termos em inglês, que foram utilizados por Othero (2009).

2 Gramática Gerativa: potencialidades para o Processamento de Linguagem Natural (PLN)

Consideramos a língua "um conjunto (finito ou infinito) de sentenças, cada sentença sendo finita em extensão e construída a partir de um conjunto finito de elementos" (CHOMSKY, 2015, p. 18). Essa concepção possibilita-nos estabelecer regras que possam separar sentenças gramaticais, que são reconhecidas como tal em uma língua, que parecem "aceitáveis a um falante nativo" (CHOMSKY, 2015, p. 18), das sentenças agramaticais, as que não são aceitáveis. Assim, podemos entender a gramática de uma língua como um "mecanismo que gera todas as sequências gramaticais" (CHOMSKY, 2015, p. 18) dessa língua. Dado que "[q]ualquer gramática de uma língua irá *projetar* o *corpus* de enunciados observados, finito e mais ou menos acidental, em um conjunto (presumivelmente infinito) de enunciados gramaticais" (CHOMSKY, 2015, p. 19, grifos do autor), pode-se dizer que "uma gramática reflete o comportamento do falante, que, baseado em uma experiência finita e acidental com a língua, pode produzir ou compreender um número indefinido de novas sentenças." (CHOMSKY, 2015, p. 19).

Quando falamos em sequências gramaticais de uma língua, falamos da perspectiva morfossintática, e não semântica. Podemos parafrasear Chomsky construindo sentenças como (1) "infinitos pássaros dormem em espelhos de açúcar" e (2) "pássaros os acima voam cabeças das dos homens". No caso de (1), ainda que a sequência não seja "dotada de sentido" ou "significativa" em um contexto não literário, pode ser considerada gramatical do ponto de vista da organização dos seus constituintes. No caso de (2), ainda que possamos, com alguma dificuldade, reorganizar os constituintes e compreender o significado do conjunto, temos uma estrutura não aceitável como está, agramatical portanto.

Chegamos, portanto, a um conceito geral de Gramática, em que cada sentença pode ser decomposta em unidades constituintes denominadas sintagmas. Essas unidades têm como núcleo as categorias gramaticais da língua e se relacionam entre si em uma estrutura de sentido que pode ser expressa em formato que se assemelha a uma árvore (Figura 1). Essas estruturas formais permitem que possamos estabelecer regras gerais para uma língua.

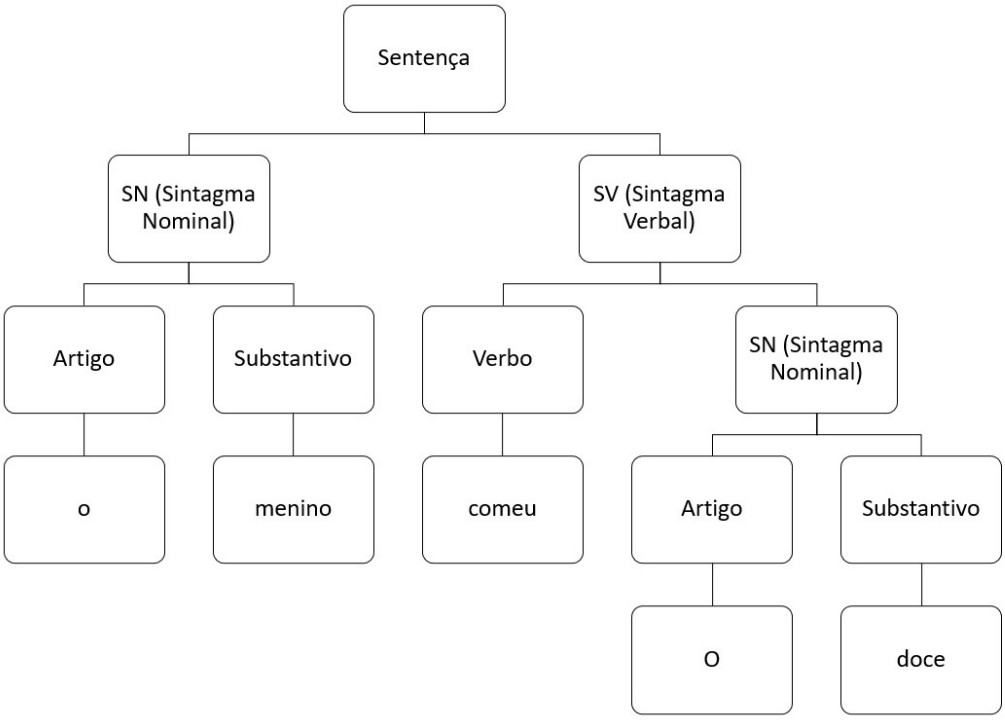

Figura 1. Diagrama arbóreo de (3) O menino comeu o doce.
Fonte: Autoria própria.

Para uma descrição mais detalhada dessa teoria, remetemos o leitor a Chomsky (2015) e, para um estudo dessas estruturas no português, para Souza e Silva e Koch (2011) e Othero (2009). O

exemplo de decomposição e de sentença que apresentaremos a seguir foi adaptado de Chomsky (2015, p. 38-39). Dada a sentença (3) "O menino comeu o doce", percebemos que as palavras não são agrupadas de modo aleatório, mas seguem regras de agrupamento que podem ser representadas do seguinte modo:

1. Sentença → SN (Sintagma nominal) + SV (Sintagma verbal)
2. SN → Artigo + Nome
3. SV → Verbo + SN
4. Artigo → o, a, os, as...
5. Nome → menino, menina, meninos, meninas, bolo, bolos, doce, doces...
6. Verbo → comeu, comeram, pegou, pegaram, comprou, compraram...

Cada regra deve ser interpretada na forma X → Y como a instrução "reescreva X como Y", de modo que teríamos como gerar, por substituição, sentenças como (4) "O menino comeu o bolo"; (5) "A menina comprou o doce"; (6) "Os meninos pegaram os bolos" e assim por diante. Poderíamos, igualmente, gerar sentenças como (7) "O bolo comeu o doce"; (8) "Os doces comeram as meninas", entre outras que seriam gramaticalmente aceitáveis, embora não o fossem semanticamente, pelo menos em contextos não literários. Podemos, ainda, determinar regras que regulam a organização dos termos no interior dos sintagmas em uma língua como o Português: artigos femininos no plural só podem estar ligados a substantivos femininos no plural, por exemplo. Também podemos determinar regras que regulam o relacionamento entre os sintagmas: caso o Sintagma Nominal fosse composto por um artigo e um substantivo no plural, o verbo teria de estar conjugado na terceira pessoa do plural, e assim por diante. O diagrama da Figura 1 seria um modo de representar uma gramática que analisa os constituintes a partir de um conjunto de regras que especificamos. Uma sequência de palavras é um constituinte se pudermos ligá-la "a um único ponto de origem" ao qual atribuímos um rótulo. Percebe-se pelo diagrama arbóreo da Figura 1 que o Sintagma Nominal aparece no nível abaixo da Sentença, ao lado do Sintagma Verbal, e que também aparece como componente do Sintagma Verbal. No primeiro caso, ele desempenha a função de sujeito: "O menino"; no segundo, de objeto direto: "o doce".

Com a finalidade de atender ao Português Brasileiro, as regras originais propostas por Chomsky (2015) precisam ser adaptadas, ainda que não alteradas em sua essência. Um excelente trabalho nesse aspecto foi desenvolvido por Othero (2009). Trata-se de uma pesquisa detalhada e consistente que serviu de base para a construção de nosso algoritmo. Na seção seguinte, vamos apresentar a Base de Dados que é utilizada por nosso *Parser* e, na Seção 4, as características técnicas e de processamento segundo as regras estudadas em Othero (2009, p. 142), com algumas adaptações e justificativas.

3 Modelagem e construção do Banco de Dados do Parser *Parsero*

Para o Processamento de Linguagem Natural, o primeiro problema é ter um léxico categorizado e atualizado, trabalho que requer pesquisa criteriosa e custosa. Encontramos esse material disponível no banco de dados DELAF_PB (MUNIZ, 2004, 2015). O DELAF_PB é um arquivo em formato "dic" que contém 9.072.146 de linhas com palavras simples e flexionadas, conforme o padrão: palavra, canônica. Classe+traços: flexão. Cada linha é uma entrada de texto terminada com um caractere de quebra de linha () e composta por até cinco campos delimitados por pontuação de acordo com o formato do *Dictionnarie Electronique du LADL* (DELA) (MUNIZ, 2004, p. 17).

Nesse formato, conforme o exemplo "fundamental,fundamental.ADJ+Pd:ms:fs", uma vírgula (",") foi utilizada para separar as duas formas de cada palavra, a flexionada e a canônica, não flexionada. Para substantivos e adjetivos, a forma canônica está atribuída ao gênero masculino, quando não se trata de palavra de dois gêneros, e, para verbos, está no infinitivo. Um ponto (".") à frente de uma palavra precede a indicação de uma categoria correspondente às classes gramaticais: substantivos, adjetivos, artigos, preposições, conjunções, numerais, pronomes, verbos, advérbios e interjeições. Também há entradas para prefixos, siglas e abreviaturas. Cada verbete pode estar categorizado em mais de uma classe gramatical; nesse caso há uma entrada para cada classe ou, no caso de verbos, a entrada pode, ainda, repetir-se para cada uma das flexões.

Um sinal de adição ("+") foi utilizado para conectar classes gramaticais a subtipos ou traços, que podem trazer informações semânticas. Por exemplo, "ART+Def", trata-se de um artigo definido; "ART+Ind", artigo indefinido; "PRO+Dem", pronome demonstrativo etc. Cada classe pode ser associada a traços de acordo com sua categoria. Os substantivos podem ser próprios ou coletivos; os numerais podem ser cardinais, ordinais, multiplicativos ou fracionários; os artigos podem ser definidos ou indefinidos; os pronomes podem ser demonstrativos, relativos, interrogativos, de tratamento, possessivos ou pessoais; as conjunções podem ser coordenativas, subordinativas ou correlativas. Esses traços das conjunções podem permitir, no futuro, construir algoritmos que possam detectar as relações de sentido entre orações.

Ao fim de cada linha do arquivo "dic", pode ou não haver um sinal de dois pontos (":"), que foi utilizado para delimitar informações de flexão, tempo, forma, valor, gênero, número, grau, pessoa. No dicionário, um verbete pode conter muitas flexões.

A implementação do Banco de Dados a partir do DELAF_PB foi feita com o Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL 12.7 (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2021) e apoiada pela ferramenta de administração Dbeaver (COMMUNITY DBEAVER, 2021). A operação de alimentar o Banco foi realizada por meio de algoritmos de conversão das entradas do DELAF-PB (MUNIZ, 2015) e foi codificada em linguagem Go 1.16.4 (THE GO AUTHORS, 2021).

Inicialmente, as entradas do dicionário foram submetidas a uma análise em busca de padrões que pudessem permitir transformar a estrutura do DELAF_PB para a modelagem de um banco de dados. Com o SGBD e por meio de *scripts* em *Structured Query Language* (SQL), foram criadas 11 tabelas (Figura 2). A tabela principal (Léxico) contém 6 atributos. Uma unidade léxica foi formada pela união de uma palavra em sua forma flexionada e não flexionada (canônica). Cada classe gramatical foi associada a um núcleo funcional descrito em Othero (2009, p. 141), que funciona como chave de identificação primária. O produto desse trabalho, o Banco de Dados populado, foi documentado e disponibilizado em (PACHECO; GUARANHA, 2021a).

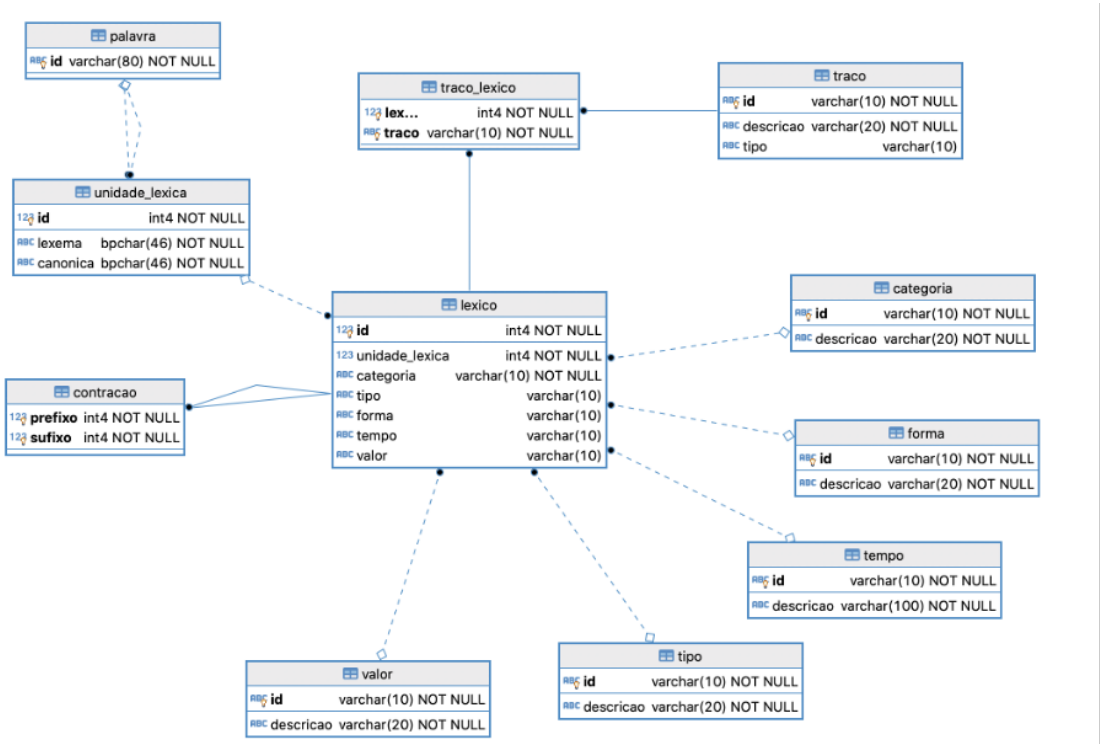

Figura 2. Modelo entidade-relacionamento do Banco de Dados.
Fonte: Autoria própria.

No processo de construção do Banco de Dados, foram atendidas algumas especificidades quanto à natureza do nosso léxico. Uma unidade do léxico como "casa", por exemplo, pode pertencer a duas

categorias gramaticais, verbo e substantivo. Caso seja verbo, a forma canônica corresponde ao verbo no infinitivo "casar"; caso seja substantivo, a forma canônica corresponde ao substantivo feminino "casa". A duplicação dentro de uma base pode ser um desperdício de memória e tomar tempo maior para a consulta de dados. Para dar conta dessa ambiguidade, foram criadas três tabelas: Palavra, Unidade Léxica e Léxico. A estratégia de criar essas três tabelas permitiu que a unidade lexical "casa", por exemplo, fosse armazenada apenas uma vez na tabela Palavra e que ela fosse só referenciada nas tabelas Unidade Léxica e Léxico (Figura 2) por meio do índice (id). Assim, na tabela Unidade Léxica, a palavra "casa" é armazenada como verbo e substantivo sem que se repita no banco.

Consideramos a palavra associada à sua forma canônica como uma unidade lexical que embasa a estrutura do nosso Banco de Dados, assim como ocorre no conjunto lexical do DELAF_PB. Para as categorias de pessoa, gênero, grau e número, manteve-se o padrão linguístico de reuni-las dentro da tabela de traço semântico (Traço). Isso está de acordo também com o que propôs Bryce-Codd (1972 apud (LAKE; CROWTHER, 2013, p. 75)) em sua "Quarta Forma normal" para padronização de dados. Essa solução permitiu contemplar a característica do nosso Banco de Dados, em que uma classe pode conter muitos traços e um traço semântico pode ser atribuído a várias classes, conforme Figura 2.

Finalmente, adotamos uma estratégia para dar conta de contrações, ou seja, de aglutinações de preposições e pronomes com outras palavras as quais geram um novo vocábulo, entradas que estão representadas no DELAF pela união de duas classes pela letra "X" (como em PREPXART para a unidade lexical "da", preposição "de" mais artigo "a"). Como nem combinações, nem contrações envolvem diretamente categorias gramaticais e, por definição, têm independência em relação a elas, a estratégia consistiu em agrupá-las em uma nova entidade com cada entrada identificada pelo prefixo e sufixo correspondentes. Isso foi feito por meio da tabela Contração (Figura 2), que recebe o sufixo e o prefixo da tabela Léxico. As categorias utilizadas foram as seguintes: PROXPRO, contração de pronome com pronome, como "aqueloutra", por exemplo; PREPXADV, contração de preposição mais advérbio, como "dali"; PREPXART, contração de preposição mais artigo, como nos casos de crase "à" e "do", no sentido de posse, como em "carro do homem"; PREPXPRO, contração de preposição mais pronome, como "nalguma", com sentido de "em alguma"; e PREXPREP, como em "dentre".

O Banco de Dados povoado contém 860.778 palavras, 879.155 unidades léxicas e um total de 1.193.295 unidades léxicas classificadas (uma unidade léxica pode pertencer a mais de uma categoria lexical). A descrição de todas as categorias gramaticais encontra-se documentada em Pacheco e Guaranha (2021a).

4 Aspectos técnicos e regras utilizadas na construção do *Parser Parsero*, um algoritmo de Processamento de Linguagem Natural

O processador sintático Parsero foi desenvolvido em linguagem GO (THE GO AUTHORS, 2021) sob o padrão MVC (Modelo-Visão-Controlador), que é uma base comum para muitos modelos baseados em Web (SOMMERVILLE, 2011, p. 108)(. O programa é uma API (Application Programming Interface) criada com a biblioteca Gorilla Mux (THE GORILLA AUTHORS, 2021). Esse padrão de projetos MVC tem três partes, assim utilizadas em nosso Parser: no Modelo, primeira parte, ficam as funções para consulta e inserção dos dados na base léxica, as quais foram descritas na seção anterior. A Visão, segunda parte, abriga um único endereço que, ao ser chamado, aciona o Controlador, terceira parte, em que estão abrigadas as instruções do algoritmo. O Controlador, então, executa as seguintes tarefas: recebe a sentença, consulta a base de dados e retorna as possibilidades de classificação para cada palavra. Depois disso, recebe esse resultado e percorre as regras de produção, as quais detalharemos nesta seção (Tabela 2), até encontrar uma válida para a sequência. No final, retorna um arquivo de texto com as unidades lexicais rotuladas de acordo com suas categorias, agrupadas em sintagmas e encapsuladas em formato (JSON) JavaScript Object Notation (ECMA-404 INTERNACIONAL, 2017).

O método escolhido para percorrer as regras foi a Análise Sintática Descendente Recursiva (ASDR) com derivação à esquerda (THAIN, 2020, p. 37). As regras de produção formam uma árvore que pode ser percorrida da raiz, a sentença, até as folhas, os núcleos funcionais. Nessa estrutura, os sintagmas são os ramos da árvore, e os nós que ficam localizados onde seriam as folhas são representados pelos

núcleos sintáticos definidos nas regras (Tabela 2). A ASDR caminha pela árvore da raiz até as folhas sempre executando primeiro o nó à esquerda. Em nossa abordagem, ela recebe um vetor com as classes possíveis para cada palavra e, ao atingir uma folha da árvore, compara cada possibilidade de classificação com o núcleo funcional da regra vigente. O ciclo é repetido até não haver mais possibilidades.

O algoritmo completo para um conjunto de regras de produção de tamanho n, em que cada possibilidade de produção é representada pelo símbolo β_i, é o seguinte:

entrada: sentença
retorno : regra válida

```
1  Função Principal(sentença):
2      escolher uma produção α tal que α → β1|β2|...|βn
3      para cada i de 1 a n faça
4          se βi é um não terminal NT então
5              invocar procedimento βi()
6          senão se (βi é um terminal Σ) e (βi == símbolo atual) então
7              avance na sentença para o próximo símbolo
8          senão
9              retorna regra inválida
10         fim
11         retorna regra inválida
12     fim
13 fim
```

Como cada regra de produção pode evocar a si mesma a cada momento, o algoritmo tem natureza recursiva. Uma sentença é considerada inválida se não pode ser expressa por nenhuma das regras de produção estabelecidas e, se há um erro, ele é classificado em erro de léxico (quando não encontra palavra no Banco de Dados), erro sintático (sentenças agramaticais) ou erro interno (evento inesperado como queda da rede, problema de acesso ao Banco de Dados etc.). A interface gráfica final foi construída com a biblioteca *React* em linguagem EcmaScript 6 (ECMA INTERNACIONAL, 2015).

Para a construção das regras de produção, utilizamos como base aquelas propostas por Othero (2009), que, por sua vez, seguem a Teoria X-Barra que é, no dizer de Chomsky (2014, p. 389-398), uma segunda fase, para além dos "primórdios da gramática da estrutura sintagmática" Chomsky (2014, p. 390). Um dos pontos destacados por Chomsky acerca dessa teoria é que

> ao invés de aparecer como nas gramáticas de estrutura sintagmática tentando construir a estrutura de cima para baixo ('*top-down*'), aqui ela se move dos itens lexicais para cima. Assume que os itens lexicais já vêm colocados em um pequeno conjunto de categorias possíveis, em que cada item em cada categoria possui um conjunto de traços que na verdade dizem o que o item lexical relevante pode fazer em uma derivação/computação. Assim, na medida em que um item lexical se 'projeta' por meio da estrutura, ele carrega seus traços com ele, e estes determinam como eles podem se combinar, onde, e como eles irão ser lidos (CHOMSKY, 2014, p. 392).

Desse modo, é possível elaborar macrorregras gramaticais que prevejam estruturas como complementos e adjuntos, por exemplo. As regras são orientadas para conseguir representar estruturas sintáticas cujo núcleo exige ou aceita complementos e regras gramaticais que possibilitem analisar sintagmas cujo núcleo não aceita ou não exige complementos. Empregando os conceitos da Teoria X-Barra, nosso *Parser* consegue lidar com sentenças como em (9) "A casa verde foi ocupada" e (10) "A casa de Maria foi ocupada", conforme Figura 3 e Figura 4 respectivamente.

O diagrama da Figura 3 representa um adjunto adnominal, e o da Figura 4 um complemento nominal. Não contemplamos, neste projeto, a distinção entre complementos e adjuntos, uma vez que os Sintagmas Preposicionais podem assumir tanto as funções de complemento nominal quanto de adjunto adnominal e que, para distinguir essas funções, teríamos de recorrer a critérios de ordem "predominantemente semântico-pragmática" (SOUZA E SILVA; KOCH, 2011, p. 25).

Em Othero (2009, p. 142), as regras de produção são formadas por um núcleo funcional único, que pode ser: Determinante (D), Substantivo (N), Verbo (V), Adjetivo (A), Advérbio (Adv), Preposição (P), Quantificador (Q), Numeral (Num). Também pode ser formado por um sintagma ou por um nó X-Barra, que pode conter ou não um complemento, por isso a notação (') aparece ao lado dele.

Nas regras definidas por Othero (2009, p. 142)[2], cada um dos sintagmas é composto por um núcleo funcional que representa o símbolo central do sintagma. Foram definidos os seguintes sintagmas: a) Nominais (SN): têm como núcleo os Substantivos (N); b) Determinantes (SD): podem conter como núcleo Artigos, Pronomes Demonstrativos, Pronomes Pessoais ou estarem ocultos na sentença. São representados como determinantes (D); c) Numerais (SNum): têm como núcleo os Numerais (Num); d) Possessivos (SPoss): têm como núcleo os Pronomes Possessivos (Poss); e) Quantificadores (SQ): têm como núcleo os quantificadores como algum, todos, cada, nenhum; f) Adjetivais (SA): têm como núcleo os Adjetivos (A); g) Adverbiais (SAdv): têm como núcleo os Advérbios (Adv); e h) Verbais (SV): têm como núcleo os Verbos (V).

Além desses sintagmas, utilizaremos, ainda, dois outros sintagmas para lidar com sentenças complexas. O primeiro, o Sintagma Flexional (SI, do *Inglês Inflectional Phrase*), dá conta das sentenças que contêm mais de um verbo em sua estrutura. Quando isso ocorre, esses dois verbos são genericamente entendidos como locuções verbais pelo nosso Parser, independentemente de serem essas locuções constituídas ou não por um verbo principal, que não se flexiona, mais um auxiliar, como em (11) "Maria vai sair" (Figura 5), ou por dois verbos principais, que são passíveis de serem flexionados, como em (12) "Maria acordou chorando" (Figura 6). Para uma discussão mais profunda sobre locuções verbais, encaminhamos o leitor para Othero (2009, p. 126-131).

Além disso, recorremos à solução de Othero (2009) para tratar estruturas com orações subordinadas conectadas por conjunções, acrescentando o SC (Sintagma Complementizador), conforme descrito pelo autor:

> A posição de núcleo do CP [SC em nosso Parser] será ocupada por **que**, **se** ou **quando** (como em João sabe se a Maria lê bastante; João sabe quando a Maria chegou). Com essas mesmas regras, podemos analisar também frases que contenham uma estrutura verbo auxiliar + verbo principal dentro do IP complemento (A Maria disse que iria chegar mais tarde) (OTHERO, 2009, p. 139, grifos do autor).

Nosso *Parser* executou corretamente essas estruturas, conforme exemplo da Figura 7, (13) "João sabe se a Maria lê bastante".

Além disso, o Sintagma Complementizador, que denominamos SC em nosso trabalho, permite que se processem frases com estruturas recursivas como em (14) "A Maria disse que João falou que estava com fome" (Figura 8).

Quanto à ordem de processamento das regras, nosso Parser adotou o método de processar primeiro as mais complexas, em que há mais do que um núcleo funcional na regra. Por exemplo, em (15) "O carro azul", opta-se primeiro pelo processamento das regras em que o substantivo (N) é precedido ou sucedido de um adjetivo (A) (regras N' $\rightarrow$ N' SA e N' $\rightarrow$ SA N'), e somente se essas regras não são válidas parte-se para a regra mais simples em que só há um substantivo (N' $\rightarrow$ N). Não fosse desse modo, em estruturas como (15), o processamento terminaria antes que todas as palavras pudessem ser avaliadas, porque a regra anterior (N' $\rightarrow$ N) já seria executada com sucesso, o adjetivo seria descartado e seguir-se-ia para o restante da sentença. Com essa mudança foi necessário também adaptar regras que poderiam resultar em recursões infinitas à esquerda, conforme exemplo na Tabela 1.

Na Tabela 1, o Sintagma Nominal apresentava recursividade infinita à esquerda na regra (N' $\rightarrow$ N' SP), que foi substituída pelas regras (N' $\rightarrow$ N N'') e (N'' $\rightarrow$ SP N''). O símbolo N'' foi criado, podendo não conter valor nenhum. Quando aplicado a todas as regras, esse raciocínio levou à formação de novos símbolos X-Barra (Tabela 2). Em alguns casos, o elemento nulo foi explicitamente usado para identificar a possibilidade de ocultar o elemento.

2 Optamos por adaptar a terminologia das regras de Othero (2009), que usa a terminologia inglesa *Phrasal*. Preferimos, como estamos trabalhando com o Português Brasileiro, a expressão Sintagma. Desse modo, usaremos SN, SV, SA, Sadv, SP e assim por diante, como Sintagma Nominal, Sintagma Verbal, Sintagma Adjetival, Sintagma Adverbial, Sintagma Preposicionado etc.

Tabela 1. Exemplo de eliminação da recursão à esquerda.

Antes	**Depois**
SN $\rightarrow$ N'	SN $\rightarrow$ N'
N' $\rightarrow$ N' SP	N' $\rightarrow$ N N''
N' $\rightarrow$ N	N'' $\rightarrow$ SP N''
	N'' $\rightarrow$ nulo

Fonte: Autoria própria.

Esse raciocínio segue o método da fatoração à esquerda, que consiste na reescrita de uma regra para adiar seu processamento até que ele seja possível (THAIN, 2020, p. 41). Por exemplo, pelas regras de Othero (2009), a frase (16) "O jogador de futebol caiu" geraria um ciclo infinito em nosso algoritmo ao tentar classificar a palavra "jogador", porque o nó N' retornaria a si mesmo (na sequência N' $\rightarrow$ N') se não fossem especificadas outras regras para que o processador não percorra novamente regras pelas quais já passou. Pelas novas regras, ao chegar ao nó N', é obrigatória a existência de um substantivo (N' $\rightarrow$ N N''). Se não for possível classificar a palavra como substantivo, o algoritmo não prossegue. Feitas essas adaptações, a Gramática do *Parsero* ficou composta pelas regras apresentadas na Tabela 2.

Nos casos em que o adjetivo precede o substantivo e que esse adjetivo também pode ser um substantivo, como em (17) "O pobre homem de Roma morreu", adotou-se a regra de prioridade para que as palavras que vêm em primeiro lugar sejam categorizadas pelo Parser como substantivos e as que vêm em segundo como adjetivos, pois essa é a forma mais comum no Português Brasileiro, que admite tanto as formas "pobre homem" como "homem pobre", diferente do Inglês, por exemplo, que admite apenas "poor man" . As Figura 3, Figura 4, Figura 5, Figura 6, Figura 7 e Figura 8, apresentadas neste trabalho, são resultados gerados pelo aplicativo.

5 Considerações finais

Este artigo apresentou a construção de um Parser Sintático para o Português Brasileiro, que utilizou o recurso disponível na Internet, o Dicionário DELAF_PB (MUNIZ, 2004) e as regras propostas por Othero (2009, p. 142), estas, por sua vez, baseadas no conceito de Gramática Gerativa, de Chomsky (2015), e expandidas pela Teoria X-Barra (CHOMSKY, 2014; OTHERO, 2009).

A partir das regras, empreendemos um intenso processo de refatoração para adaptá-las ao algoritmo utilizado (Algoritmo Descendente Recursivo), o que nos levou à criação de novas regras que eliminaram os ciclos recursivos à esquerda, facilitando o processamento.

A base léxica no formato DELA foi adaptada para o formato SQL e disponibilizada em código aberto. O projeto do Banco de Dados respeita o formalismo de Bryce-Codd (1972 apud (LAKE; CROWTHER, 2013, p. 75)), e evitou duplicações para poupar memória de processamento. A estrutura proposta permitiu também que novas palavras pudessem ser inseridas sem classificação, o que priorizou a expansão do léxico. Essa estrutura deixa aberta a possibilidade de que posteriormente as palavras possam ser classificadas manualmente por usuários ou por métodos matemáticos apreendidos a partir da base já existente.

A tela de interação com o usuário foi desenvolvida em linguagem GO, sob o padrão MVC, Model (modelo) View (visão) e Controller (Controle), padrão que facilita a troca de informações entre a interface do usuário e os dados no banco, fazendo com que as respostas sejam mais rápidas e dinâmicas. Para processar as sentenças simples e compostas do Português Brasileiro, sem uso de pontuação, foi gerada uma API que as analisa e exibe a árvore sintática por meio de uma interface gráfica.

Além de ter propiciado reflexões sobre as potencialidades da Gramática Gerativa para o desenvolvimento de Processadores de Linguagem Natural, notadamente para atender às sofisticadas construções sintáticas do Português Brasileiro, este trabalho também constituiu uma oportunidade de pesquisa interdisciplinar que buscou aproximar as ciências da linguagem e as ciências da computação. Além

Tabela 2. Regras de Produção para o *Parser Parsero*.

Sintagma Inflexional	Sintagma Numeral	Sintagma Preposicional
SI → SD I'	SNum → Num	SP → P'
SI → I'	SNum → Num SP;	P' → SAdv P'
I' → I" SV		P' → P SD
I" → I I"	Sintagma Possesivo	P' → P SAdv
I" → SD I"	SPoss → Poss SN	P' → P SC
I" → nulo	SPoss → SN Poss	P' → P SP
	SPoss → Poss SNum;	P' → P;
Sintagma Complementizador	Sintagma Quantificador	Sintagma Adverbial
SC → C SI;	SQ → Q SD	AdvP → Adv'
Sintagma Determinante	SQ → Q SP	Adv' → Adv SP
SD → D SN	SQ → SD Q;	Adv' → Adv Adv"
SD → D SPoss		Adv" → Adv' Adv"
SD → D SNum	Sintagma Adjetival	Adv" → nulo;
SD → SN	SA → A'	Sintagma Verbal
SD → SPoss	A' → SAdv A'	SV → V'
SD → SNum	A' → A SP	V' → SAdv V'
SD → D	A' → A SC	V' → V SD
	A' → A A"	V' → V SP
Sintagma Nominal	A" → SAdv A"	V' → V SC
SN → N'	A" → SP A"	V' → V SA
N' → N SP	A" → nulo;	V' → V SAdv
N' → N SC		V' → V SI
N' → N N"		V' → V V"
N' → AP N'		V" → SAdv V"
N" → SC N"		V" → PP V"
N" → SP N"		V" → nulo;
N" → SA N"		
N" → nulo;		

Fonte: Adaptação de Othero (2009, p. 142).

disso, disponibilizar os resultados, tanto a base de dados quanto o algoritmo, é um estímulo para que outros pesquisadores possam aperfeiçoar ou possam criar outros processadores sintáticos que serão a base para sistemas inteligentes os quais possam interagir com escritores ou falantes do Português.

Referências

CHOMSKY, N. *Ciência da linguagem*. São Paulo: Editora UNESP, 2014.

CHOMSKY, N. *Estruturas sintáticas*. São Paulo: Vozes, 2015.

COMMUNITY DBEAVER. *Dbeaver: Free Universal Database Manager*. 2021. Disponível em: https://dbeaver.io/download/.

ECMA INTERNACIONAL. *ECMAScriptLanguage Specification*. 2015. Disponível em: https://262.ecma-international.org/6.0/. Acesso em: 16 out. 2021.

ECMA-404 INTERNACIONAL. *The JSON Data Interchange Syntax*. 2017. Disponível em: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/. Acesso em: 16 out. 2021.

LAKE, P.; CROWTHER, P. *Concise guide to databases: a practical introduction*. 1st edition. New York, NY: Springer London, 2013. (Undergraduate topics in computer science).

MUNIZ, M. C. M. *A construção de recursos linguístico-computacionais para o português do Brasil: o projeto de Unitex-PB*. 2004. Dissertação de Mestrado – Instituto de Ciências Matemáticas de São Carlos, USP. Disponível em: http://ladl.univ-mlv.fr/brasil/bibliografia/oto/DissMuniz2004.pdf.

MUNIZ, M. C. M. *DELAF-PB: Dicionário de Palavras Simples Flexionadas para o Português Brasileiro*. [S.l.: s.n.], 2015. Disponível em: http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html. Acesso em: 4 mai. 2022.

OTHERO, G. de A. *A gramática da frase em português: algumas reflexões para a formalização da estrutura frasal em português*. Porto Alegre: EDIPUCRS, 2009. Disponível em: https://bibliodigital.unijui.edu.br:8443/xmlui/handle/123456789/1490. Acesso em: 16 out. 2021.

PACHECO, W. E. A.; GUARANHA, M. F. *Banco de dados para análise sintática em sentenças do português brasileiro*. 2021. Disponível em: https://github.com/Kiriwill/FATEC-IPI-ParserDB. Acesso em: 16 out. 2021.

PACHECO, W. E. A.; GUARANHA, M. F. *Parser-api*. 2021. Disponível em: https://github.com/Kiriwill/parser-api. Acesso em: 4 mai. 2022.

PARSERO: parser sintático para o português brasileiro. 2021. Disponível em: https://parserov1.herokuapp.com/. Acesso em: 4 mai. 2022.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA E SILVA, C. P. de; KOCH, I. V. *Linguística aplicada ao português: sintaxe*. São Paulo: Cortez, 2011.

THAIN, D. *Introduction to compilers and language design*. 2. ed. [S.l.]: Independently Published, 2020.

THE GO AUTHORS. *GoLang Versão 1.16.4*. 2021. Disponível em: https://pkg.go.dev/runtime. Acesso em: 16 out. 2021.

THE GORILLA AUTHORS. *Gorilla Web Toolkit*. 2021. Disponível em: https://github.com/gorilla/mux. Acesso em: 16 out. 2021.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL Database Management System*. 2021. Disponível em: https://www.postgresql.org/download/. Acesso em: 16 out. 2021.

Contribuições dos autores

Manoel Francisco Guaranha: Conceituação, Metodologia, Validação, Supervisão, Escrita – revisão e edição; **Willian Emerson Afonso Pacheco**: Metodologia, Programas, Escrita – rascunho original.

A Apêndice

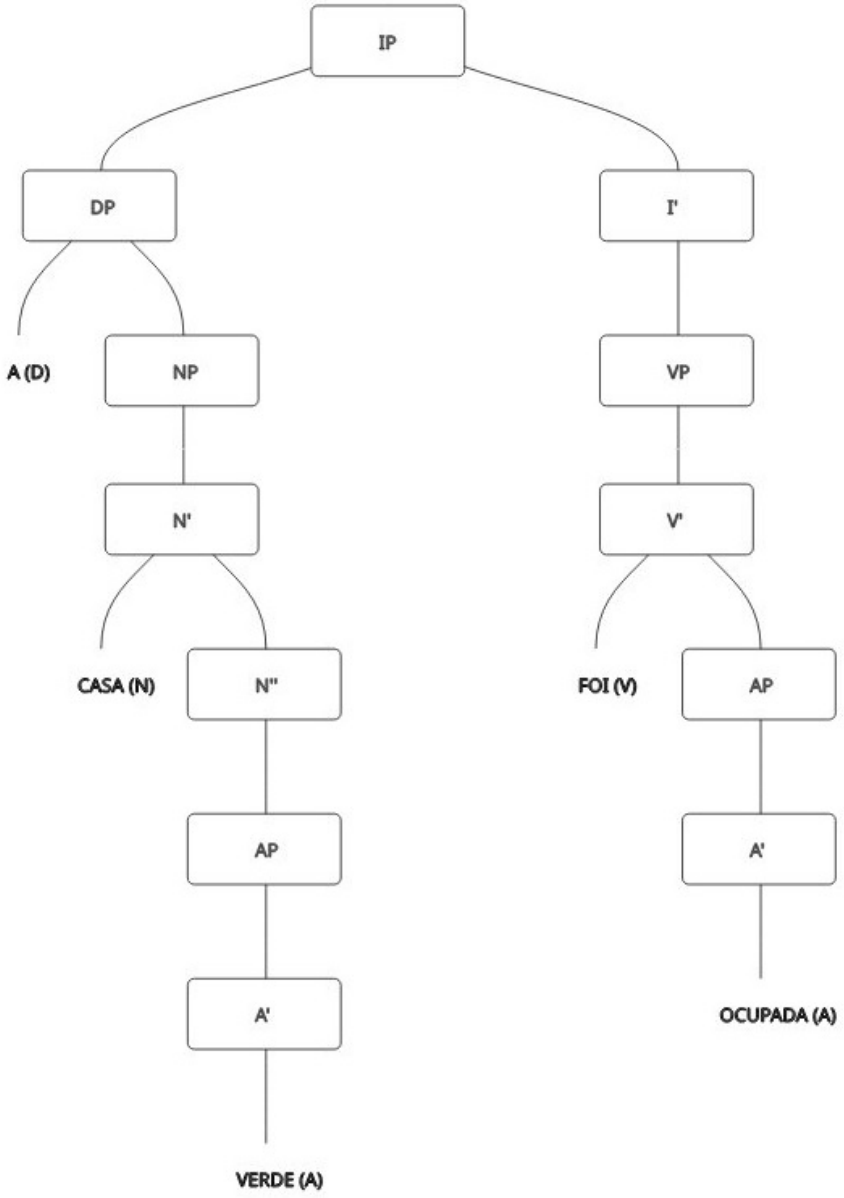

Figura 3. Diagrama arbóreo de Sintagmas Nominais – Teoria X-Barra – sentença: "A casa verde foi ocupada" (9).

Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.

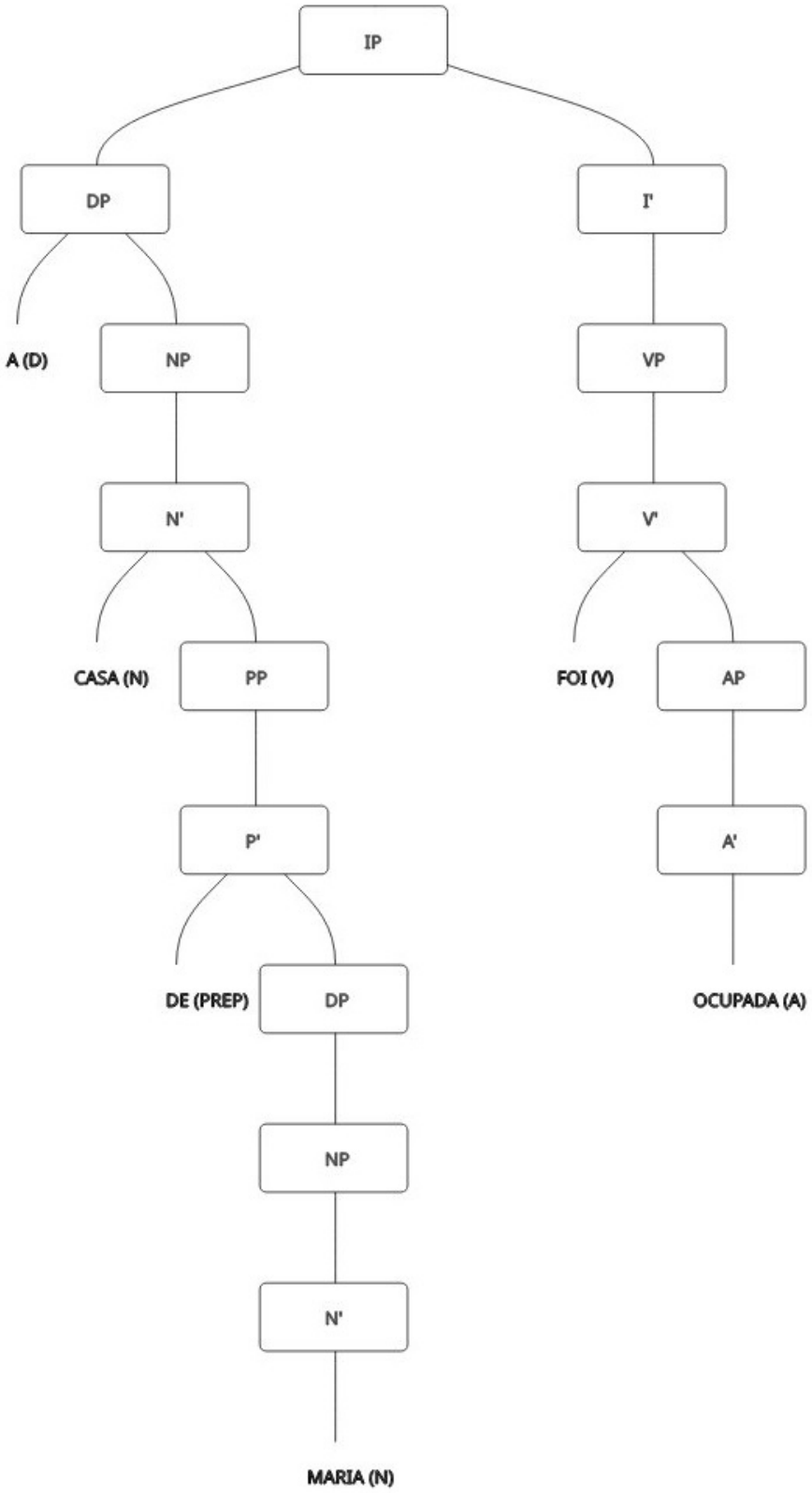

Figura 4. Diagrama arbóreo de Sintagmas Nominais – Teoria X-Barra – sentença: "A casa de Maria foi ocupada" (10).

Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.

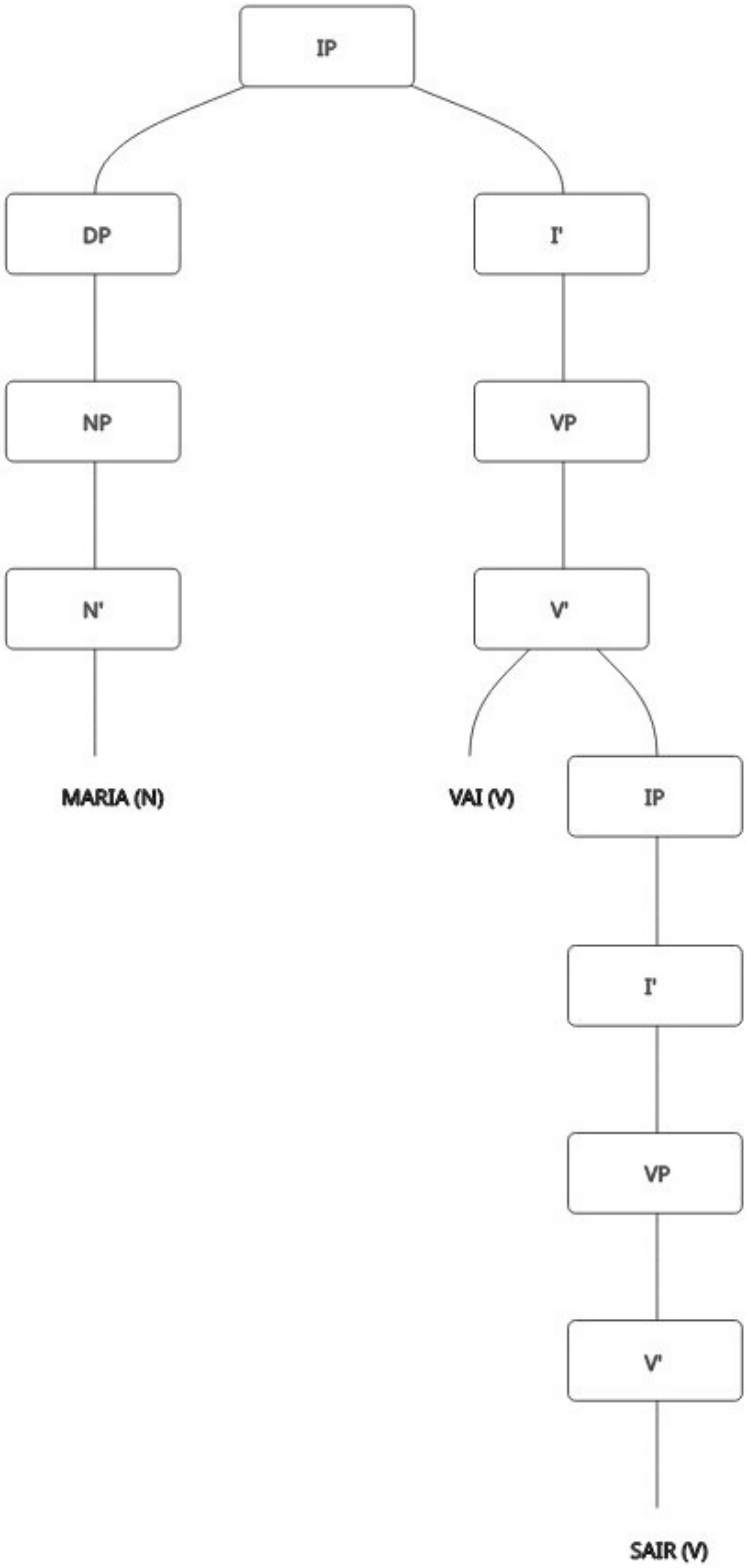

Figura 5. Sentença com verbo auxiliar: "Maria vai sair" (11).
Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.

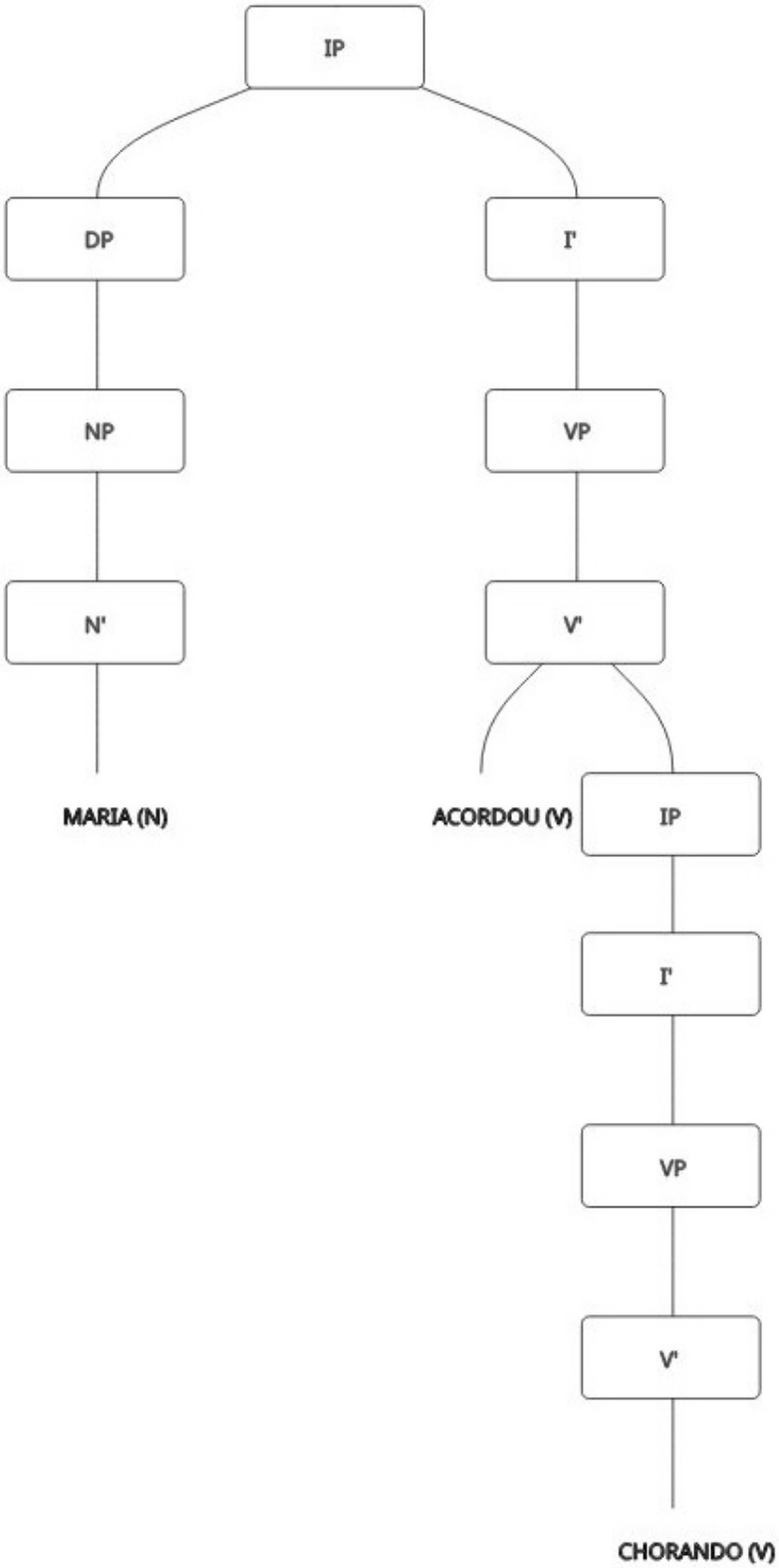

Figura 6. Sentença com dois verbos principais: "Maria acordou chorando" (12).
Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.

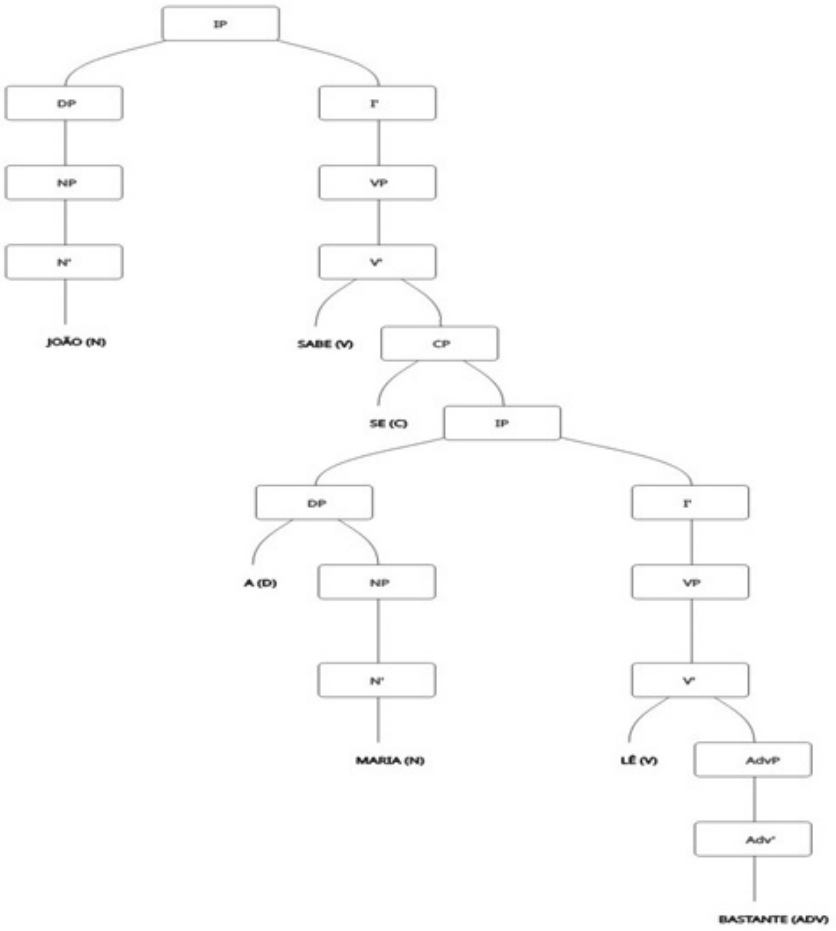

Figura 7. Exemplo de sentença com Sintagma Complementizador – "João sabe se a Maria lê bastante" (13).
Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.

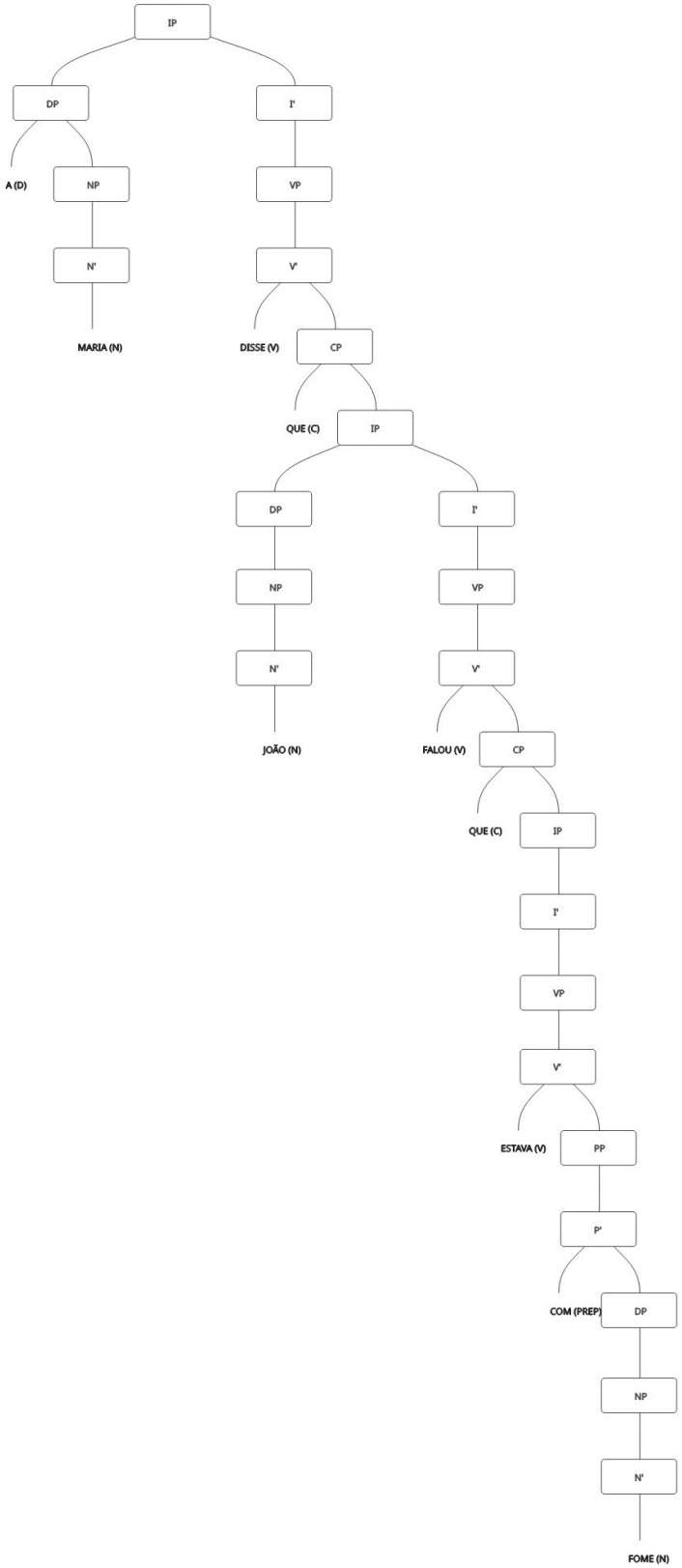

Figura 8. Exemplo de sentença com estrutura recursiva – sentença: "A Maria disse que João falou que estava com fome" (14).

Fonte: Autoria própria. Interface gráfica do *Parsero* 1.0.