UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

VERIFICA-UD: UMA FERRAMENTA ONLINE PARA VERIFICAÇÃO DE TEXTOS EM PORTUGUÊS ANOTADOS NO FORMATO CoNLL-U SEGUNDO O PADRÃO *UNIVERSAL DEPENDENCIES*

LUCELENE LOPES
MAGALI SANCHES DURAN
THIAGO ALEXANDRE SALGUEIRO PARDO

Nº 444

RELATÓRIOS TÉCNICOS

São Carlos – SP
Ago./2023

Natural Language Processing initiative (NLP2) of the Center for Artificial Intelligence (C4AI) of the University of São Paulo, sponsored by IBM and FAPESP

POeTiSA

POrtuguese processing – Towards Syntactic Analysis and parsing

Verifica-UD - Uma ferramenta Online para verificação de textos em Português anotados no formato CoNLL-U segundo o padrão *Universal Dependencies*

Lucelene Lopes, Magali Sanches Duran,

Thiago Alexandre Salgueiro Pardo

Agosto/2023

Relatório Técnico do
Núcleo Interinstitucional de Linguística Computacional (NILC)

Sumário

1. Introdução

Este relatório descreve o Verifica-UD, uma ferramenta online para verificar possíveis problemas em sentenças anotadas segundo o padrão *Universal Dependencies* - UD (de Marneffe et al., 2021; Nivre et al., 2020) no formato CoNLL-U de acordo com as diretrizes UD para o português estabelecidas no projeto POeTiSA no decorrer da construção do córpus Porttinari-base (Pardo et al., 2021; Duran et al., 2023).

A ferramenta Verifica-UD tem o potencial de impulsionar a produção de córpus anotados em português, pois permite detectar erros automaticamente, aliviando parte do trabalho dos anotadores que revisam a anotação automática. Por trabalhar com regras, o Verifica-UD também pode contribuir para uma maior homogeneização da anotação UD empreendida pela comunidade brasileira de Processamento de Linguagem Natural. O Verifica-UD é gratuito e, de acordo com experimentos descritos por Lopes et al. (2023), apresenta um bom desempenho na detecção de problemas. Além disso, a ferramenta disponibiliza páginas de ajuda detalhadas que facilitam a correção manual dos problemas.

O Verifica-UD implementa um total de regras que resultam na detecção de 106 tipos de erros e 22 tipos de avisos divididos em três níveis: estrutural, morfossintático e de relações de dependência. A ferramenta pode ser acessada pelos endereços http://verificaud.icmc.usp.br e http://verificaud.icmc.usp.br:24080/verificaud/, assim como pelo portal web do projeto POeTiSA (https://sites.google.com/icmc.usp.br/poetisa).

Esse documento apresenta uma descrição detalhada dos erros e avisos detectados em cada um dos três níveis: nível estrutural (Seção 2), nível morfossintático (Seção 3) e nível de relações de dependências (Seção 4). Em seguida, na Seção 5, apresentamos em linhas gerais a ferramenta online Verifica-UD, com ênfase na sua interface de utilização. Finalmente, fazemos considerações finais e descrevemos os trabalhos futuros imaginados para a ferramenta.

2. Nível Estrutural

Inicialmente, o nível estrutural verifica a integridade do arquivo CoNLL-U submetido. Nesse sentido, é importante definir como consideramos que o arquivo deve ser.

O formato CoNLL-U é um formato de texto para representar a anotação sintática de sentenças em uma linguagem natural. Ele é usado para representar córpus anotados com o padrão *Universal Dependencies*.

Cada sentença no formato CoNLL-U é um bloco de linhas composto por:

- meta-informações (linhas com o caractere `#` na primeira posição) e
- linhas de tokens (linhas que descrevem o corpo da sentença).

O bloco correspondente a cada sentença não pode ter linhas em branco, pois uma linha em branco (linha sem caracteres) é o indicador de término de um bloco de sentença.

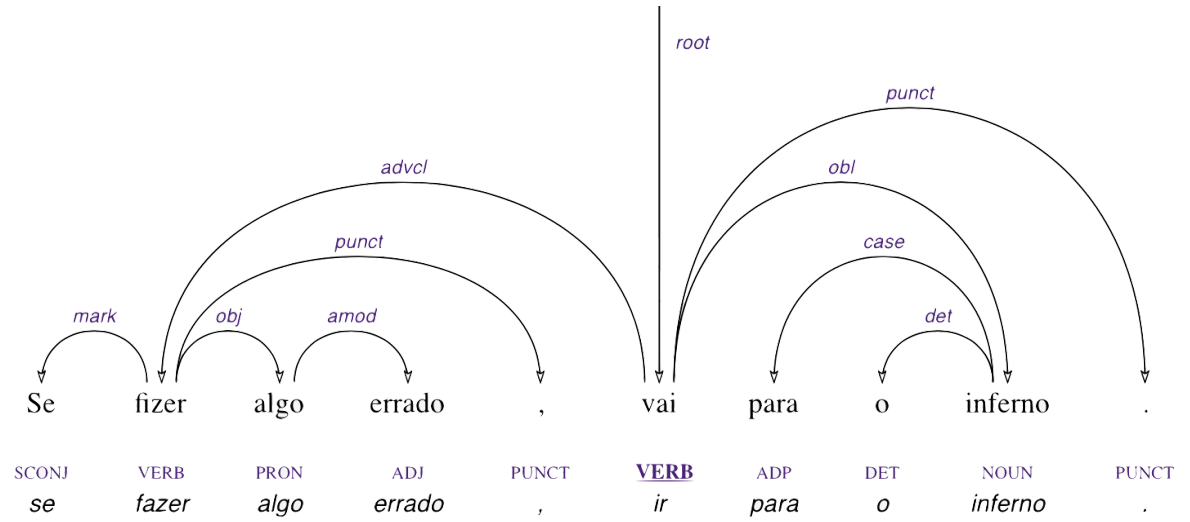
Um bloco de sentença pode ter várias linhas de meta-informação, porém cada sentença deve ter duas informações essenciais:

- um identificador de sentença definido com a meta-informação:
    - `# sent_id = <`***string***`>`
- o texto original da sentença definido com a meta-informação:
    - `# text = <`***string***`>`

Após as linhas de meta-informação, as linhas do corpo da sentença devem representar um token por linha, com os seguintes campos:

- ID: o identificador do token na sentença (um número sequencialmente atribuído a partir do número 1);
- FORM: o token na sua forma utilizada na sentença;
- LEMMA: o lema ao qual o token corresponde;
- POS: a etiqueta morfossintática (PoS - *Part of Speech*) associada ao token.
- XPOS: a etiqueta PoS estendida associada ao token (campo não utilizado no decorrer da construção do córpus Porttinari-base até o momento);
- FEAT: os atributos morfológicos do token;
- HEAD: o ID do token head da relação de dependência do token;
- DEPREL: a relação de dependência do token com seu token head;
- DEPS: a relação de *enhanced dependency* do token (campo não utilizado no decorrer da construção do córpus Porttinari-base até o momento);
- MISC: informações adicionais sobre o token.

Por exemplo, uma sentença simples como "*Se fizer algo errado, vai para o inferno.*" pode ser anotada no formato CoNLL-U conforme apresentado na Figura 1, que mostra tanto o formato CoNLL-U como o desenho da árvore de dependência correspondente. Vale salientar que a sentença "Se fizer algo errado, vai para o inferno" pode ter também uma interpretação equivalente a "Se fizer errado algo". Neste caso, "errado" seria anotado como ADV advmod porque estaria modificando o próprio predicado e não o objeto.




```
# sent_id = FOLHA_DOC009367_SENT031
# text = Se fizer algo errado, vai pro inferno.
1   Se  se  SCONJ   _   _   2   mark    _   _
2   fizer   fazer   VERB    _   Mood=Sub|Number=Sing|Person=3|Tense=Fut|VerbForm=Fin    6   advcl   _   _
3   algo    algo    PRON    _   Gender=Masc|Number=Sing|PronType=Ind    2   obj _   _
4   errado  errado  ADJ _   Gender=Masc|Number=Sing|VerbForm=Part   3   amod    _   SpacesAfter=No
5   ,   ,   PUNCT   _   _   2   punct   _   _
6   vai ir  VERB    _   Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   0   root    _   _
7-8 pro _   _   _   _   _   _   _   _
7   para    para    ADP _   _   9   case    _   _
8   o   o   DET _   Definite=Def|Gender=Masc|Number=Sing|PronType=Art   9   det _   _
9   inferno inferno NOUN    _   Gender=Masc|Number=Sing 6   obl _   SpacesAfter=No
10  .   .   PUNCT   _   _   6   punct   _   _
```

Figura 1. Exemplo de formato CoNLL-U para uma sentença e árvore de dependência correspondente.

Nota-se que, no exemplo da Figura 1, existe uma contração ("*pro*") que corresponde a dois tokens ("*para*" e "*o*"). De acordo com as diretivas da UD, uma contração deve ter os tokens que a compõem incluídos. No formato CoNLL-U, as contrações devem ser mantidas em uma linha individual seguidas pelas linhas dos tokens que as compõem, os quais devem ser etiquetados conforme suas funções na sentença. Observa-se que a linha da palavra contraída não possui etiquetas associadas a ela. A numeração da contração deve indicar a numeração do intervalo de tokens que a compõem. No exemplo da Figura 1, a contração "*pro*" recebe o identificador 7-8, pois ela corresponde ao token com o identificador 7 - "*para*", e ao token com identificador 8 - "*o*".

2.1. Verificação Estrutural no Verifica-UD

A verificação estrutural no Verifica-UD começa pela conferência da integridade do formato CoNLL-U de cada sentença a fim de garantir que as meta-informações obrigatórias (`# sent_id` e `# text`) sejam preenchidas e que todas as demais linhas anteriores ao corpo da sentença possuam o caractere `#` na primeira posição. Além disso, verifica-se que não existam linhas indevidas entre as linhas de

meta-informação e entre as linhas de tokens. Os erros detectados pelo Verifica-UD nesse primeiro momento são os erros S01 a S05 (vide Tabela 1).

O segundo momento da verificação estrutural é a conferência da integridade das linhas do corpo da sentença que devem ter, cada uma, 10 campos separados por caracteres de tabulação ("`\t`"). Além disso, nenhum dos 10 campos pode estar vazio, pois, pela definição do formato CoNLL-U, um campo sem informações deve ser ocupado pelo caractere sublinhado ("_"). O erro detectado nesse segundo momento é o erro S06 (vide Tabela 1). Todos os erros S01 a S06 interrompem a verificação dos demais erros, pois o arquivo não possui o formato mínimo para ser tratado como um arquivo CoNLL-U.

O terceiro momento da verificação estrutural é a análise dos tokens da sentença, que devem corresponder ao *string* na meta-informação do texto original tanto pela forma do token (campo FORM) quanto pela informação de espaços após cada token (menção a `SpaceAfter` no campo MISC). Dessa forma, as linhas relativas aos tokens da sentença devem permitir a reconstrução da sentença original informada na linha de meta-informação obrigatória que começa por `# text`, bem como possuir os identificadores (campo ID) corretamente numerados em ordem crescente a partir de 1. Os erros detectados nesse primeiro momento são os erros S07 a S12 (vide Tabela 1).

O quarto e último momento da verificação estrutural é a conferência da integridade da árvore de dependência da sentença. Para isso, é necessário que a árvore tenha uma única raiz (etiqueta DEPREL `root`) e apresente todos os demais tokens da sentença conectados a ela. Os erros dessa natureza detectados pelo Verifica-UD são:

- Índice do token no campo HEAD ausente ou incorreto: cada token em uma árvore de dependência deve ter um token HEAD, que é o token do qual ele depende. O Verifica-UD verifica casos em que o identificador do token no campo HEAD indica um identificador inválido.
- Relações de dependência incorretas: cada token em uma árvore de dependência deve ter uma relação de dependência, que indica sua relação com seu token HEAD. O Verifica-UD verifica casos em que um token não possui uma relação de dependência válida.
- Token sem caminho de relações de dependência até o token raiz (`root`) da sentença. Note que, se houver um ciclo de dependência para um conjunto de tokens, eles não terão acesso à raiz da sentença (por exemplo, quando dois tokens tiverem indicação de dependência de um para o outro e vice-versa).
- Token com um identificador de HEAD duplo (por exemplo, identificador de uma palavra contraída no campo HEAD).

Os erros detectados nesse quarto e último momento são os erros S13 a S16 (vide Tabela 1). Na verdade, todos os erros do nível estrutural têm o código iniciando com a letra S, remetendo à palavra inglesa *structural*.

Tabela 1. Lista de erros da verificação estrutural.

Código	Mensagem
S01	Sentença sem identificador (verificação interrompida).
S02	Sentença sem texto original (verificação interrompida).
S03	Linha inesperada - meta-informação sem # (verificação interrompida).
S04	Linha inesperada entre linhas de tokens (verificação interrompida).
S05	Erro de formato desconhecido (verificação interrompida).
S06	Token sem os 10 campos obrigatórios.
S07	Sentença original incompatível com os tokens informados.
S08	Numeração errada de tokens que compõem contração (token inexistente).
S09	Numeração errada de tokens que compõem contração (números errados).
S10	Numeração errada de tokens que compõem contração (números fora de sequência).
S11	Numeração errada de tokens (números fora de sequência).
S12	Numeração errada de tokens (relação de dependência com token inexistente).
S13	Múltiplos tokens marcados como root na sentença (token {} também marcado como root).
S14	Sentença sem token root.
S15	Token sem caminho até o token root da sentença.
S16	Token com mais do que uma dependência.

3. Nível Morfossintático

Para melhor entender as verificações do nível morfossintático, é importante ter em mente o conjunto de etiqueta PoS definido pela UD e como ele é usado na anotação do português. Mais detalhes sobre a anotação de etiquetas PoS para o português podem ser encontrados no Manual de Anotação de PoS tags (Duran, 2021; Duran et al., 2022). As 17 etiquetas PoS definidas pela UD são descritas a seguir:

- **ADP**, adposições, uma classe fechada que, em português, corresponde às preposições, como "de", "para" e "com";
- **ADJ**, adjetivos, uma classe aberta que inclui palavras como "bonitas", "último" e "vermelha"; os numerais ordinais escritos por extenso, como "primeiro", "centésima" e "duodécimo"; os numerais ordinais expressos em dígitos como "20º" ou "13ª"; e formas de verbos no particípio como "cansado" e "adequado ";
- **ADV**, advérbios, uma classe aberta com um subconjunto fechado, os advérbios primitivos (aqueles não formados com o sufixo "-mente"). Exemplos do subconjunto fechado são as palavras "cedo", "agora" e "acima". Exemplos do subconjunto aberto são as palavras "normalmente" e "insanamente". Também são incluídas formas abreviadas desses advérbios que aparecem em expressões como "social e economicamente" onde o advérbio "social" é usado como uma forma abreviada do advérbio "socialmente";
- **AUX**, verbos auxiliares e de cópula, uma classe fechada de verbos gramaticalizados, ou seja, que possuem função gramatical. No decorrer da construção do córpus Porttinari-base, essa classe engloba todas as conjugações dos verbos "ser", "estar", "haver", "ir", "ter" e "vir";
- **CCONJ**, conjunções coordenativas, uma classe fechada que contém, por exemplo, "e", "mas" e "portanto";
- **DET**, determinantes, uma classe fechada que inclui artigos como "um" e "o", bem como pronomes que estejam modificando substantivos, por exemplo, "cujo" em "cujo pai", "aquele" em "aquele problema" e "meus" em "meus amigos";
- **INTJ**, as interjeições, uma classe aberta que inclui, por exemplo, "tchau", "oi" e "poxa";
- **NOUN**, substantivos, uma classe aberta que inclui palavras como "presidente**, "**quartos**, "**bandeirinha**, "**salões" e **"**bola**;**
- **NUM**, numerais cardinais, uma classe aberta que inclui os numerais cardinais escritos por extenso, como "duas", "trinta" e "quinhentos", bem como todos os números escritos com dígitos como "51", "-3.1415", "¾", "1,5" e até datas como "25/12/1974";
- **PART**, partículas, uma classe que não é utilizada no decorrer da construção do córpus Porttinari-base;

- **PRON**, pronomes, uma classe fechada atribuída aos pronomes substantivos como "eu", "mim", "aquilo" ou pronomes em função substantiva (função de sujeito ou objeto de verbos) como "aqueles" em "aqueles que vieram" ou "estes" em "prefiro estes;
- **PROPN**, nomes próprios , uma classe aberta que contempla denominações. No córpus Porttinari, definiu-se que a classe compreende todos os nomes de pessoas, empresas, órgãos governamentais, projetos, locais, títulos de obras, etc. Esses nomes podem ser simples ou compostos e normalmente são escritos em maiúsculas, como "João", "Paris", "SENAI", "Senado Federal", "Sem Parar" (exceção: iPhone);
- **PUNCT**, todas as pontuações, por exemplo, ponto final, exclamação, interrogação, vírgula, dois pontos, ponto-e-vírgula, aspas e parênteses;
- **SCONJ**, conjunções subordinativas, uma classe fechada que contém, por exemplo, "conquanto", "se" e "segundo";
- **SYM**, todos os símbolos simples e compostos, por exemplo, "R$", "US$" e "%";
- **VERB**, verbos, uma classe aberta que inclui todas as conjugações dos verbos plenos, por exemplo, "canta", "jogar", "chorastes" e "terias";
- **X**, tudo que não pertence ao vocabulário da língua, como as palavras estrangeiras "petit-four" e "bullying", bem como onomatopéias como "oinc" e "crec".

Quanto aos lemas e atributos morfológicos possíveis, o Verifica-UD se baseia na definição feita para um léxico de português específico para anotação em UD, o PortiLexicon-UD (Lopes et al., 2022). Esse recurso define, para 862.322 itens lexicais, suas possíveis etiquetas PoS e seus respectivos lemas e atributos morfológicos. No entanto, o PortiLexicon-UD não cobre as etiquetas **PROPN**, **PUNCT**, **SYM** e **X**, além de, obviamente, não conter a totalidade dos itens lexicais das classes **ADJ**, **ADV**, **INTJ**, **NOUN**, **NUM** e **VERB,** as quais, por serem classes abertas, não possuem um número finito de itens. Dessa forma, palavras das classes abertas podem ser aceitas desde que apresentem um conjunto de etiquetas de atributos morfológicos compatível com a sua respectiva etiqueta de PoS. Um relato mais completo das possibilidades de anotação de atributos morfológicos pode ser encontrado em Lopes et al. (2023).

3.1. Verificação Morfossintática no Verifica-UD

A verificação morfossintática do Verifica-UD foca nos tokens individuais de uma sentença, observando se suas etiquetas PoS, lemas e atributos morfológicos estão de acordo com as diretrizes para UD em português definidas no projeto POeTiSA no decorrer da construção do córpus Porttinari-base.

O processo de verificação morfossintática começa com a observação da etiqueta PoS, pois, se a etiqueta for uma das previstas para o item lexical no PortiLexicon-UD, o processo de verificação é feito com respeito aos respectivos lema e atributos morfológicos previstos no léxico. No entanto, caso a etiqueta PoS não exista no PortiLexicon (classes PROPN, PUNCT, SYM e X), apenas uma verificação da compatibilidade do lema e dos atributos morfológicos possíveis é feita. A compatibilidade de lema e atributos morfológicos foi definida anteriormente em (Lopes et al., 2022) quando da construção do PortiLexicon-UD.

A grande maioria dos itens lexicais encontrados em córpus do português é verificada através do PortiLexicon-UD. No caso de itens lexicais não encontrados no léxico, apesar de apresentarem PoS cobertas pelo léxico, o Verifica-UD confere se suas etiquetas de atributos morfológicos são compatíveis com a PoS e, se forem, não gera uma mensagem de erro, mas apenas um aviso de que a palavra é desconhecida.

Dessa forma, o nível morfossintático detecta 29 erros possíveis, anotados com códigos que vão de T01 a T29 e 14 avisos com códigos que vão de t30 a t43. A letra T é utilizada pois esses erros e avisos são referidos no Verifica-UD como problemas de *Tagging*, já que correspondem a problemas que são feitos usualmente na atribuição de etiquetas (*tags*) morfossintáticas. A Tabela 2 apresenta mensagens de erro e avisos detectados nesse nível.

Tabela 2. Lista de mensagens de erro e avisos da verificação morfossintática.

Código	Mensagem
T01	Contração não pode ter informações de lema, PoS e atributos morfológicos.
T02	Nenhum token ADP não abreviado pode ter atributos morfológicos.
T03	Todo token ADP abreviado só pode ter atributo morfológico 'Abbr=Yes'.
T04	Nenhum token CCONJ não abreviado pode ter atributos morfológicos.
T05	Todo token CCONJ abreviado só pode ter atributo morfológico 'Abbr=Yes'.
T06	Nenhum token SCONJ não abreviado pode ter atributos morfológicos.
T07	Todo token SCONJ abreviado só pode ter atributo morfológico 'Abbr=Yes'.
T08	Nenhum token INTJ não abreviado pode ter atributos morfológicos.
T09	Todo token INTJ abreviado só pode ter atributo morfológico 'Abbr=Yes.

T10	Nenhum token X não abreviado pode ter atributos morfológicos, exceto 'Foreign=Yes'.
T11	Todo token X abreviado só pode ter atributo morfológico 'Abbr=Yes' ou 'Foreign=Yes'.
T12	Nenhum token ADV não abreviado pode ter atributos morfológicos.
T13	Todo token ADV abreviado só pode ter atributo morfológico 'Abbr=Yes.
T14	Todo token NUM textual deve ter atributo morfológico 'NumType=Card/Frac', e pode ter apenas o atributo morfológico 'Gender=Fem/Masc'.
T15	Todo token NUM numérico deve ter lema igual ao token na sua forma utilizada na sentença.
T16	Todo token NUM em forma de dígitos deve ter atributo morfológico 'NumType=Card'.
T17	Todo token NOUN pode ter apenas os atributos morfológicos 'Gender=Fem/Masc', 'Number=Sing/Plur' e 'Abbr=Yes'.
T18	Todo token ADJ pode ter apenas os atributos morfológicos 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Abbr=Yes', 'VerbForm=Part' e 'NumType=Ord'.
T19	Todo token VERB ou AUX deve ter atributos morfológicos 'VerbForm=Inf/Ger/Part/Fin' e, de acordo com o tempo verbal, pode ter apenas os atributos morfológicos 'Mood= =Ind/Sub/Cnd/Imp', 'Tense=Pres/Past/Fut/Imp/Pqp', 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Person=1/2/3', 'Voice=Pass' e 'Abbr=Yes'.
T20	Todo token DET deve ter atributos morfológicos 'PronType=Art/Dem/Ind/Rel/Int/Prs' e, de acordo com a opção, pode ter apenas os atributos morfológicos 'Definite=Def/Ind', 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Person=1/2/3' e 'Poss=Yes'.
T21	Todo token PRON deve ter atributos morfológicos 'PronType=Dem/Ind/Rel/Int/Prs' e, de acordo com a opção, pode ter apenas os atributos morfológicos 'Case=Acc/Dat/Nom', 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Person=1/2/3' e 'Poss=Yes'.
T22	Todo token PROPN não deve ter atributos morfológicos.
T23	Todo token PROPN deve ter lema igual ao token na sua forma utilizada na sentença.
T24	Todo token PUNCT não deve ter atributos morfológicos.
T25	Todo token PUNCT deve ter lema igual ao token na sua forma utilizada na sentença.

T26	Todo token SYM não deve ter atributos morfológicos.
T27	Todo token SYM deve ter lema igual ao token na sua forma utilizada na sentença.
T28	A PoS tag PART não está sendo utilizada para a anotação.
T29	A PoS tag {} não faz parte das etiquetas válidas para UD.
t30	Token ADP (classe fechada) preposição desconhecida.
t31	Token AUX (classe fechada) verbo auxiliar ou de cópula desconhecido.
t32	Token CCONJ (classe fechada) conjunção coordenativa desconhecida.
t33	Token SCONJ (classe fechada) conjunção subordinativa desconhecida.
t34	Token DET (classe fechada) determinante desconhecido.
t35	Token PRON (classe fechada) pronome desconhecido.
t36	Token ADJ (subconjunto fechado) número ordinal escrito por extenso desconhecido.
t37	Token ADV (subconjunto fechado) advérbio primitivo desconhecido.
t38	Token NUM (subconjunto fechado) número cardinal escrito por extenso desconhecido.
t39	Token NOUN (classe aberta) substantivo desconhecido.
t40	Token ADJ (classe aberta) adjetivo desconhecido.
t41	Token ADV (classe aberta) advérbio desconhecido.
t42	Token INTJ (classe aberta) interjeição desconhecida.
t43	Token VERB (classe aberta) verbo desconhecido.

4. Nível de Relações de Dependência

A verificação de relações de dependência no Verifica-UD aplica uma série de regras que definem combinações de relações de dependência entre si e com informações morfossintáticas de acordo com as definições de anotação de relações de dependência estabelecidas para o português no projeto POeTiSA (Duran, 2022; Duran et al., 2022). Essas regras foram concebidas no decorrer da construção do córpus Porttinari-base (Pardo et al., 2021; Duran et al., 2023).

A lista completa das 34 etiquetas DEPREL consideradas pelo Verifica-UD é apresentada a seguir.

- ***acl*** - oração adnominal - ocorre entre um nominal (NOUN, PROPN, PRON, NUM, ADV ou ADJ), e uma oração que o modifica. Essa relação acontece da palavra modificada em direção ao predicado da oração modificadora. Até onde observamos, é uma relação que acontece da esquerda para a direita. Seu uso pode ser feito como acl apenas ou com a sub-relação acl:relcl para as orações adjetivas desenvolvidas, também chamadas de orações relativas, pois contêm um pronome relativo que remete ao termo anterior que qualificam ou especificam.
- ***advcl*** - oração adverbial - liga o predicado de uma oração matriz ao predicado de uma oração que a modifica, acrescentando-lhe informações circunstanciais de causa, tempo, conformidade, concessão, comparação, condição, consequência, finalidade, modo e proporção. As orações modificadoras circunstanciais, dependentes da ***advcl***, correspondem, em sua maioria, às orações adverbiais das gramáticas tradicionais. Essa relação pode ocorrer tanto da direita para esquerda, quanto da esquerda para a direita.
- ***advmod*** - modificador adverbial - liga uma palavra de conteúdo a advérbio (ou expressão ***fixed*** com função adverbial) que a modifica. O HEAD da relação ***advmod*** é majoritariamente constituído de verbos, adjetivos ou advérbios, porém admite também nominais. Essa relação pode ocorrer tanto da direita para esquerda, quanto da esquerda para a direita.
- ***amod*** - modificador adjetivo - liga um nominal ao adjetivo que o qualifica ou que o classifica. Essa relação pode ocorrer nos dois sentidos, embora seja mais comum da esquerda para a direita.
- ***appos*** - modificador apositivo - ocorre entre dois nominais que têm o mesmo referente extralinguístico, então o dependente serve para definir, nomear ou descrever o HEAD (por exemplo, siglas e expressões, nomes próprios, codinomes e suas descrições). Essa relação tem sentido fixo, da esquerda para a direita.

- ***aux*** - verbo auxiliar - liga um verbo auxiliado a seu auxiliar, logo todo HEAD de ***aux*** é um verbo (VERB) e o dependente é um verbo auxiliar (AUX). Essa relação tem sentido fixo, da direita para a esquerda. Seu uso pode ser feito como ***aux*** apenas ou com a sub-relação ***aux:pass*** quando o auxiliar é de voz passiva.
- ***case*** - marcador de caso - liga uma palavra de conteúdo a uma preposição (ADP) que a introduz, podendo a palavra de conteúdo estar etiquetada como NOUN, PROPN, PRON, NUM, ADV ou ADJ. Essa relação tem sentido fixo, da direita para a esquerda.
- ***cc*** - conjunção - ocorre entre um elemento coordenado e a conjunção que o introduz numa coordenação, sendo também utilizada para anotar marcadores discursivos que não têm a contrapartida de uma coordenação. Essa relação tem sentido fixo, da direita para a esquerda.
- ***ccomp*** - complemento oracional fechado - é usada para anotar um dos dois tipos de oração com valor de complemento verbal (o outro tipo utiliza ***xcomp***), sendo a relação ***ccomp*** usada para orações que admitem a expressão de um sujeito próprio (embora possa estar elíptico). Essa relação tem sentido fixo, da esquerda para a direita. Seu uso pode ser feito como ***ccomp*** apenas ou com a sub-relação ***ccomp:speech*** que é utilizada para discurso direto relatado, e nesse caso, a relação ***ccomp:speech*** pode ocorrer tanto da direita para esquerda, quanto da esquerda para a direita.
- ***conj*** - coordenado - ocorre entre dois elementos coordenados; havendo vários elementos coordenados, o primeiro deles é HEAD de uma relação ***conj*** com cada um dos demais coordenados. Essa relação tem sentido fixo, da esquerda para a direita.
- ***cop*** - verbo de cópula - liga um verbo de cópula a um predicativo (ADJ, NOUN, PROPN, ADV, PRON, NUM, SYM ou X). No projeto POeTiSA, no decorrer da construção do córpus Porttinari-base, e nos demais projetos de anotação UD em português, apenas os verbos *ser* e *estar* podem funcionar como verbos de cópula. Essa relação pode ocorrer tanto da direita para esquerda, quanto da esquerda para a direita.
- ***csubj*** - sujeito oracional - é utilizada para anotar o sujeito que se apresenta sob forma de oração, sendo o HEAD da relação ***csubj*** o predicado da oração matriz e o dependente, o predicado da oração subordinada (oração-sujeito). Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita. Seu uso pode ser feito como ***csubj*** apenas ou com a sub-relação ***csubj:pass***, quando o sujeito oracional for sujeito de uma construção passiva, ou ainda com a sub-relação ***csubj:outer***, quando a oração matriz tem dois sujeitos: um interno e um externo, mesmo se o sujeito da oração predicativa (matriz do ***csubj:outer***) estiver elíptico ou for inexistente.

- ***det*** - determinante - liga um nominal ao seu determinante (necessariamente DET). Essa relação pode ocorrer nos dois sentidos, embora seja mais comum da direita para a esquerda.
- ***discourse*** - discurso - é usada para anotar interjeições e outros elementos que não têm uma relação clara com a estrutura sintática da sentença, exceto de maneira expressiva; logo, a função dos dependentes de ***discourse*** é pragmática e não sintática. Essa relação pode ocorrer tanto da direita para esquerda quanto da esquerda para a direita.
- ***dislocated*** - deslocado - é utilizada para anotar um elemento que repete um dos papéis sintáticos já preenchidos na oração (sujeito, verbo, complemento), ou seja, um elemento redundante; portanto, aplica-se principalmente a textos com características de oralidade. Essa relação pode ocorrer tanto da direita para a esquerda quanto da esquerda para a direita.
- ***expl*** - expletivo - é usada para relacionar partículas expletivas, ou seja, sem um papel sintático (dependente de ***expl***), a um predicado (HEAD); apenas pronomes são previstos como dependentes de ***expl*** na UD. Essa relação pode ocorrer tanto da direita para a esquerda quanto da esquerda para a direita. Seu uso pode ser feito como ***expl*** apenas ou com a sub-relação ***expl:pass***, quando o expletivo é utilizado para formar a voz passiva sintética, ou seja, a voz passiva em que o verbo principal não está no particípio e que não admite a expressão de um agente da passiva.
- ***fixed*** - expressão fixa - usada para relacionar tokens que compõem uma multipalavra com função de preposição, conjunção coordenativa ou subordinativa, advérbio e pronomes, ou seja, funções de classes fechadas. Essa relação tem sentido fixo, da esquerda para a direita, partindo sempre do primeiro token em direção a cada um dos tokens seguintes que fazem parte da expressão fixa.
- ***flat*** - relação plana - foi criada para ligar elementos que têm relação não hierárquica entre si, pois não há um HEAD e um dependente naturalmente identificáveis, já que todos estão no mesmo plano, por exemplo, a relação entre os tokens de um número composto escrito por extenso. Essa relação tem sentido fixo, da esquerda para a direita, sendo sempre do primeiro token em direção a cada um dos tokens seguintes que fazem parte da expressão plana. Seu uso pode ser feito como ***flat*** apenas ou com a sub-relação ***flat:foreign*** quando os elementos ligados correspondem a uma expressão em língua estrangeira, ou ainda com a sub-relação ***flat:name*** quando os elementos ligados correspondem a um nome próprio.
- ***goeswith*** - tokens que vão juntos - é usada para anotar partes de uma palavra que foram grafadas separadamente, em desacordo com as normas ortográficas (por exemplo: “guarda chuva” sem hífen), ou tokenizadas indevidamente.

- ***iobj*** - objeto indireto - é usada para anotar a relação entre um predicado verbal e um terceiro argumento (o primeiro é ***nsubj*** e o segundo é ***obj***). Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita. Em português, apenas PRON é admitido como dependente de ***iobj***.
- ***list*** - lista - ocorre entre elementos pertencentes a uma lista, dois a dois, sempre partindo do primeiro elemento (HEAD) em direção a cada um dos demais elementos da lista, exceto se for possível utilizar uma relação ***conj***. Essa relação tem sentido fixo, da esquerda para a direita.
- ***mark*** - marcador de subordinação - é usada para ligar o predicado da oração subordinada ao marcador que a introduz, usualmente uma conjunção subordinativa (SCONJ), uma preposição (ADP), ou ainda uma expressão fixa. Essa relação tem sentido fixo, da direita para a esquerda.
- ***nmod*** - modificador nominal - liga um nominal a um outro nominal que o modifica. Essa relação pode ocorrer nos dois sentidos, embora seja majoritariamente da esquerda para a direita. Seu uso pode ser feito como ***nmod*** apenas ou com a sub-relação ***nmod:lmod*** quando for um modificador de lugar, ou ainda com a subrelação ***nmod:tmod*** quando for um modificador de tempo.
- ***nsubj*** - sujeito - utilizada para anotar a relação entre o predicado (verbal ou nominal) e o sujeito. Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita. Seu uso pode ser feito como ***nsubj*** apenas ou com a sub-relação ***nsubj:pass*** quando o sujeito for utilizado em uma construção passiva, ou ainda com a sub-relação ***nsubj:outer*** quando houver um segundo sujeito, externo à oração predicativa, mesmo quando o sujeito da oração predicativa estiver elíptico ou for inexistente.
- ***nummod*** - modificador numérico - liga um nominal a um modificador numérico (numerais cardinais por extenso ou em algarismos), portanto, todo dependente de ***nummod*** deve estar anotado como NUM. Essa relação pode ocorrer nos dois sentidos, embora seja majoritariamente da esquerda para a direita.
- ***obj*** - objeto direto - é usada para anotar a relação entre um predicado verbal e o segundo argumento (o primeiro é ***nsubj*** e o terceiro é ***iobj***). Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita. Seu uso pode ser feito como ***obj*** apenas ou com a sub-relação ***obj:agent*** quando o objeto for utilizado em uma construção passiva.
- ***obl*** - oblíquo - é usada para ligar um verbo, um adjetivo ou advérbio (HEAD da relação) a um complemento oblíquo que o modifica (dependente da relação), sendo frequentemente introduzido por preposição. Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita.

- ***orphan*** - órfão - liga dois elementos que ficaram órfãos de HEAD, em função da elipse do HEAD que tinham em comum, logo, essa relação é usada tipicamente quando há elipse de um predicado e pelo menos duas palavras de conteúdo que se ligariam a esse predicado. Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita.
- ***parataxis*** - parataxe - é uma relação entre dois elementos da sentença que poderiam ter relação sintática entre si, porém essa relação não está explicitada, podendo frequentemente ser utilizada para ligar orações (ao contrário da relação ***discourse*** que é puramente pragmática). Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita.
- ***punct*** - pontuação - liga uma palavra de conteúdo (HEAD) a um símbolo de pontuação (dependente que necessariamente deve estar anotado como PUNCT). Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita.
- ***reparandum*** - disfluência - ocorre entre dois tokens, sendo que o segundo (HEAD) corrige a ocorrência do primeiro (dependente). Essa relação tem sentido fixo, da direita para a esquerda.
- ***root*** - raiz - marca o início da árvore de dependência, ou seja, é uma relação artificial que dá início às demais relações de dependência, portanto, apenas um token em cada sentença pode ser marcado como ***root***. Essa relação parte do elemento vazio em direção ao predicado mais importante da sentença.
- ***vocative*** - vocativo - é usada para marcar o participante do diálogo a quem se dirige a mensagem, portanto, o HEAD da relação, normalmente, é o predicado da oração principal. Essa relação pode ocorrer tanto da direita para a esquerda, quanto da esquerda para a direita.
- ***xcomp*** - complemento oracional aberto - é usada para anotar um dos dois tipos de oração com valor de complemento verbal (o outro tipo utiliza ***ccomp***). Usada para orações que não admitem a expressão de um sujeito próprio, ou seja, o predicado dependente de ***xcomp*** nunca será HEAD de ***nsubj*** ou ***csubj***. Essa relação tem sentido fixo, da esquerda para a direita.

4.1. Verificação de Relações de Dependência no Verifica-UD

A definição das regras para verificação de relações de dependências é fruto de um trabalho prático de anotação de um córpus jornalístico, o Porttinari-base (Duran et al., 2023), que foi desenvolvido ao longo de dois anos.

O nível de relações de dependência detecta 61 erros, anotados com códigos que vão de P01 a P61 e 8 avisos com códigos que vão de p62 a p69. A letra P é utilizada pois esses erros e avisos são referenciados no Verifica-UD como erros de *Parsing*, já que a atribuição de relações de dependência é usualmente feita durante o processo de análise sintática. A Tabela 3 apresenta mensagens de erros e alertas de possíveis erros nesse nível.

Tabela 3. Lista de mensagens de erro e avisos da verificação de relações de dependência.

Código	Mensagem
P01	Todo token dependente de det é DET (o inverso não é verdadeiro).
P02	Todo token dependente de amod é ADJ (o inverso não é verdadeiro).
P03	Todo token dependente de iobj é PRON, tem atributo morfológico Case=Dat e lema: ['me', 'te', 'se', 'lhe', 'nos', 'vos', 'lhes'] (o inverso não é verdadeiro).
P04	Todo token dependente de obj que é PRON, tem atributo morfológico Case=Acc e lema ['me', 'te', 'se', 'o', 'a', 'nos', 'vos', 'os', 'as', 'lo', 'la', 'los', 'las', 'no', 'na', 'nos', 'nas'].
P05	Nenhum ADP é head de relação - Exceções: pode ser head da relação fixed (e se for de fixed, pode ser de punct, cc e conj também), pode ser head de conj se o dependente for ADP, pode ser head de cc se o token ADP for dependente de conj.
P06	Nenhum token CCONJ é head de relação - Exceção: pode ser head da relação fixed (e se for de fixed, pode ser de punct também), pode ser head de conj dependente de CCONJ.
P07	Nenhum SCONJ é head de relação - Exceções: pode ser head da relação fixed (e se for de fixed, pode ser de punct, cc e conj também), pode ser head de conj se o dependente for SCONJ, pode ser head de cc se o token SCONJ for dependente de conj.
P08	Nenhum DET é head de relação - Exceções: pode ser head da relação fixed (e se for de fixed, pode ser de punct, cc e conj também), pode ser head de conj se o dependente for DET, pode ser head de cc se o token DET for dependente de conj.
P09	Um token AUX não pode ser head de relação (exceto punct) se for dependente de aux, aux:pass ou cop.
P10	Nenhum token dependente de fixed é head de relação.
P11	Todo token head de case só pode ser PROPN, NOUN, PRON, ADJ, NUM, X, INTJ, SYM ou ADV.

P12	Nenhum token SCONJ pode ser dependente de nsubj, nsubj:pass, nsubj:outer, csubj, csubj:pass, csubj:outer e obj.
P13	Todo token dependente de cop só pode ter lema 'estar' ou 'ser'.
P14	Dois ou mais tokens dependentes de case não podem apontar para um mesmo token head.
P15	Nenhum token PRON pode ser dependente de mark, exceto se for head de fixed.
P16	Todo token dependente de aux só pode ter lema = 'ser', 'estar', 'ter', 'haver', 'ir' e 'vir' caso seja vir+Ger ou vir+'a'+Inf.
P17	Todo token que é ponto final não pode ter como head um token que não é head de outras relações.
P18	Todo token dependente de nummod só pode ser head de advmod, nummod, flat, conj e case.
P19	Todo token VERB dependente de xcomp nunca é head de uma relação de sujeito: nsubj, nsubj:outer e nsubj:pass.
P10	Nenhum token ADJ pode ser head de det.
P21	Todo token dependente de aux, aux:pass ou cop tem que ter ser AUX (o inverso não é verdadeiro).
P22	Todo token VERB não pode ser head de case.
P23	Todo token head de obl só pode ser VERB, ADJ, ADV ou X. Exceção: também pode ser AUX se o verbo estiver elíptico (o token head de AUX não é VERB).
P24	Todo token dependente de advmod é ADV, ADJ ou qualquer token que seja head de fixed.
P25	Todo token dependente de nummod é NUM (o inverso não é verdadeiro).
P26	Todo token dependente de nmod é PROPN, NOUN, NUM, INTJ, SYM, X ou PRON (o inverso não é verdadeiro).
P27	Todo token PUNCT é dependente de punct e vice-versa.
P28	Todo token ADP que não é head de fixed e cujo token head é um verbo no infinitivo (VerbForm=Inf) deve ser dependente de mark.
P29	Nenhum token PROPN ou NOUN pode ser dependente de appos de um token PROPN ou NOUN. Exceção: se o token head for separado por pontuação do token dependente.
P30	Nenhum token head de case pode ser dependente de obj.

P31	Nenhum token head de case pode ser dependente de iobj.
P32	Todos tokens 'muito', 'muitos', 'muita', 'muitas' que forem ADJ devem ser dependentes de amod. Exceção: caso seja dependente de xcomp ou head de cop.
P33	Todos tokens 'pouco', 'poucos', 'pouca', 'poucas' que forem ADJ devem ser dependentes de amod. Exceção: caso seja dependente de xcomp ou head de cop.
P34	Todos tokens 'onde', 'quando', 'como' que forem ADV, e não forem head de cop ou fixed, devem ser dependentes de advmod, fixed ou conj (se estiverem coordenadas com outro ADV).
P35	Todo token dependente de expl só pode ser PRON.
P36	Nenhum token head de cop é VERB.
P37	Não deve haver Deprel projetivas (cruzamento de arcos) para nenhum token.
P38	Uma sentença só pode ter um token root e ele deve ser bem formado (head 0, Deprel 'root')
P39	Nenhum token é head de mais de um nsubj.
P40	Nenhum token é head de mais de um nsubj:pass.
P41	Nenhum token é head de mais de um nsubj:outer.
P42	Nenhum token é head de mais de um csubj.
P43	Nenhum token é head de mais de um obj.
P44	Nenhum token é head de mais de um xcomp.
P45	Nenhum token é head de mais de um ccomp.
P46	Nenhum token é head de mais de um case.
P47	Todo token dependente de appos tem seu head à esquerda.
P48	Todo token dependente de mark tem seu head à direita.
P49	Todo token dependente de case tem seu head à direita.
P50	Todo token dependente de fixed tem seu head à esquerda.
P51	Todo token dependente de flat:name tem seu head à esquerda.
P52	Nenhum token pode ser simultaneamente head de nsubj e head de nsubj:pass, nsubj e csubj ou nsubj:pass e csubj.
P53	Todo token dependente de flat:foreign tem seu head à esquerda.

P54	Todo token SCONJ só pode ser dependente de mark, fixed, conj ou discourse. Exceção: se for a palavra 'quanto' pode ser dependente de advcl.
P55	Todo token head de nsubj:pass deve ter atributo morfológico Voice=Pass. Exceção: se o verbo for head de expl:pass.
P56	Todo token head de obl:agent deve ter atributo morfológico Voice=Pass.
P57	Todo token head de aux:pass deve ter atributo morfológico Voice=Pass.
P58	Todo token head e dependente de flat:foreign deve ser X.
P59	Todo token head e dependente de flat:name deve ser PROPN ou X.
P60	Todo token head e dependente de flat deve ser NUM.
P61	Todo token head de aux ou aux:pass, que não for simultaneamente head de cop, deve ser VERB.
p62	Normalmente, dois ou mais tokens dependentes de cop não podem apontar para um mesmo token head.
p63	Normalmente, dois ou mais tokens dependentes de mark não podem apontar para um mesmo token head.
p64	Normalmente, todo token dependente de flat:name só pode ser head de cc, case, det, punct, expl, nmod (se o dependente for um número cardinal - NUM NumType=Card) ou amod (se o dependente for um número ordinal - ADJ NumType=Ord).
p65	Normalmente, um token dependente de nmod só pode ter como head tokens PROPN, NOUN, NUM, SYM, X e PRON.
p66	Normalmente, todo token PUNCT final tem como head o token root.
p67	Normalmente, pontuações pareadas (aspas, parênteses, colchetes, chaves) tem o mesmo head.
p68	Normalmente, nenhum token pode ser simultaneamente head de obj e head de ccomp.
p69	Normalmente, nenhum token ADJ pode ser head ou dependente de appos.

5. Ferramenta Online Verifica-UD

A ferramenta Verifica-UD foi implementada em dois módulos:

- um programa em linguagem Python que implementa o servidor usando o modelo API REST (Richardson and Ruby, 2007), e
- uma interface web implementada com tecnologia HTML/CSS/PHP.

A operação básica da ferramenta consiste em fazer upload de um arquivo CoNLL-U (*.conllu*) que é repassado ao servidor para ser verificado automaticamente. A Figura 2 apresenta um printscreen da tela mostrando a entrada do arquivo para *upload*.

Figura 2. Tela inicial do Verifica-UD com upload de arquivo.

Assim que o arquivo é informado, inicia-se um *upload* do arquivo para a máquina servidora que executa o programa de verificação e retorna uma mensagem de sucesso quando a verificação tiver sido completada, ou uma mensagem de falha caso tenha havido algum problema de comunicação com o servidor. A Figura 3 apresenta a tela do Verifica-UD informando a mensagem de sucesso da comunicação com o servidor e processamento do arquivo.

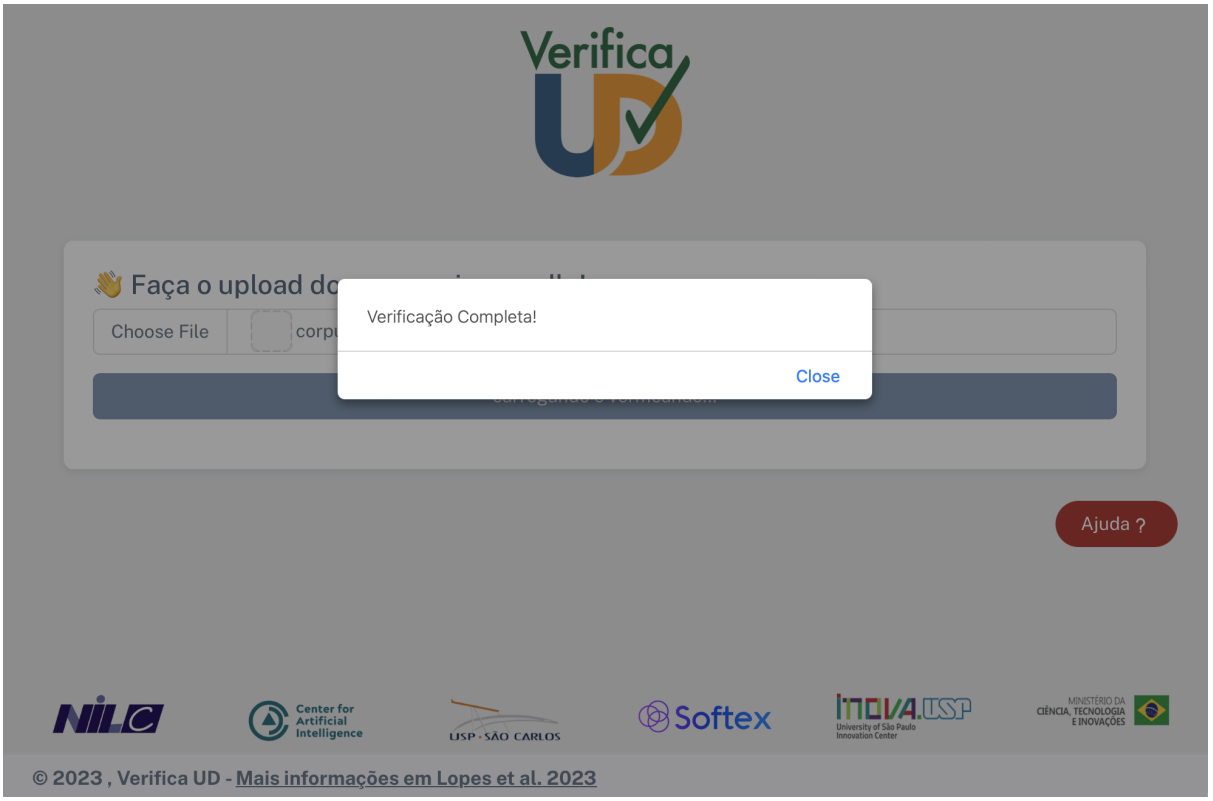


Figura 3. Tela do Verifica-UD informando sucesso do upload e verificação do arquivo CoNLL-U.

Uma vez que o arquivo tenha sido verificado e a resposta do servidor tenha sido recebida, uma tela exibirá a lista de erros e avisos encontrados devidamente agrupados por sentenças em ordem alfabética do identificador da sentença, conforme ilustrado na Figura 4.

No exemplo da Figura 4, está sendo informado que 12 possíveis erros foram encontrados. Mostra-se também uma lista com os identificadores das 5 sentenças, de um total de 7 sentenças no córpus em análise, onde os possíveis erros foram encontrados.

O usuário pode clicar em cada um desses identificadores para ver a lista de problemas encontrados na sentença. No exemplo mostrado na Figura 4, a sentença **FOLHA_DOC009367_SENT024** apresenta três erros e um aviso:

- Um erro de Parsing para o token 1;
- Um erro de Tagging para o token 1;
- Um erro de Parsing para o token 4;
- Um aviso de Parsing para o token 6.

A Figura 5 apresenta o CoNLL-U da mesma sentença onde os três erros e o aviso indicados na Figura 4 podem ser observados.

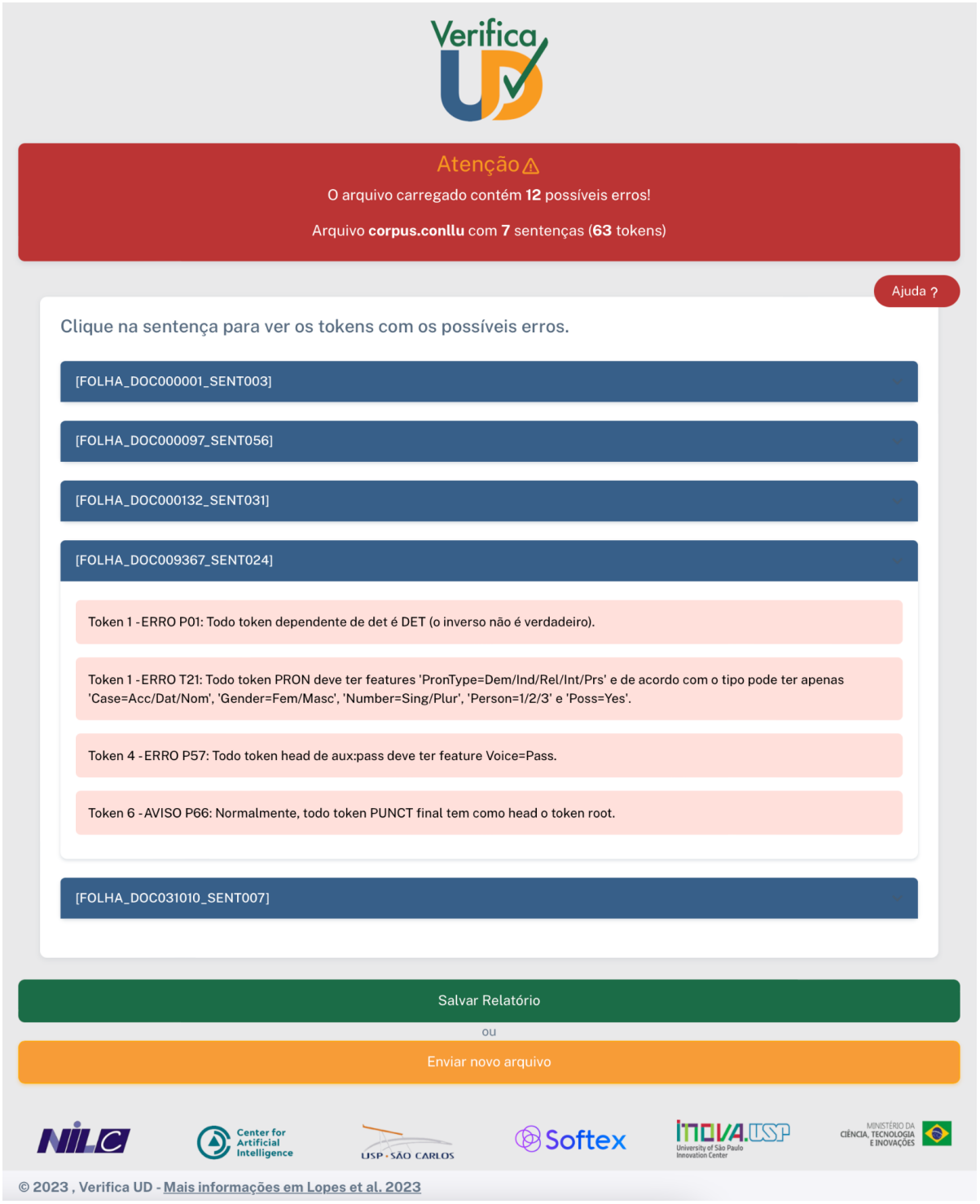


Figura 4. Tela do Verifica-UD apresentando o resultado da verificação.

```
# sent_id = FOLHA_DOC009367_SENT024
# text = A gente educa sendo educado.
1   A   o   PRON    _   Definite=Def|Gender=Fem|Number=Sing|PronType=Art    2   det _   _
2   gente   gente   NOUN    _   Gender=Fem|Number=Sing   3   nsubj   _   _
3   educa   educar  VERB    _   Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   0   root    _   _
4   sendo   ser AUX _   VerbForm=Ger    5   aux:pass    _   _
5   educado educar  VERB    _   Gender=Masc|Number=Sing|VerbForm=Part   3   advcl   _   SpaceAfter=No
6   .   .   PUNCT   _   _   5   punct   _   _
```

Figura 5. Representação CoNLL-U da sentença FOLHA_DOC009367_SENT024.

A ferramenta Verifica-UD é uma ferramenta útil para revisores humanos de córpus anotados segundo o padrão UD. Ao identificar e corrigir problemas de anotação, os revisores podem melhorar a qualidade do córpus. Nesse sentido, o Verifica-UD oferece uma opção de ajuda em que o usuário pode aprender mais sobre a anotação de textos em português em UD. As Figuras 6 e 7 apresentam telas da opção de ajuda para as opções de Tagging e Parsing de anotação em UD para o português presentes no Verifica-UD.

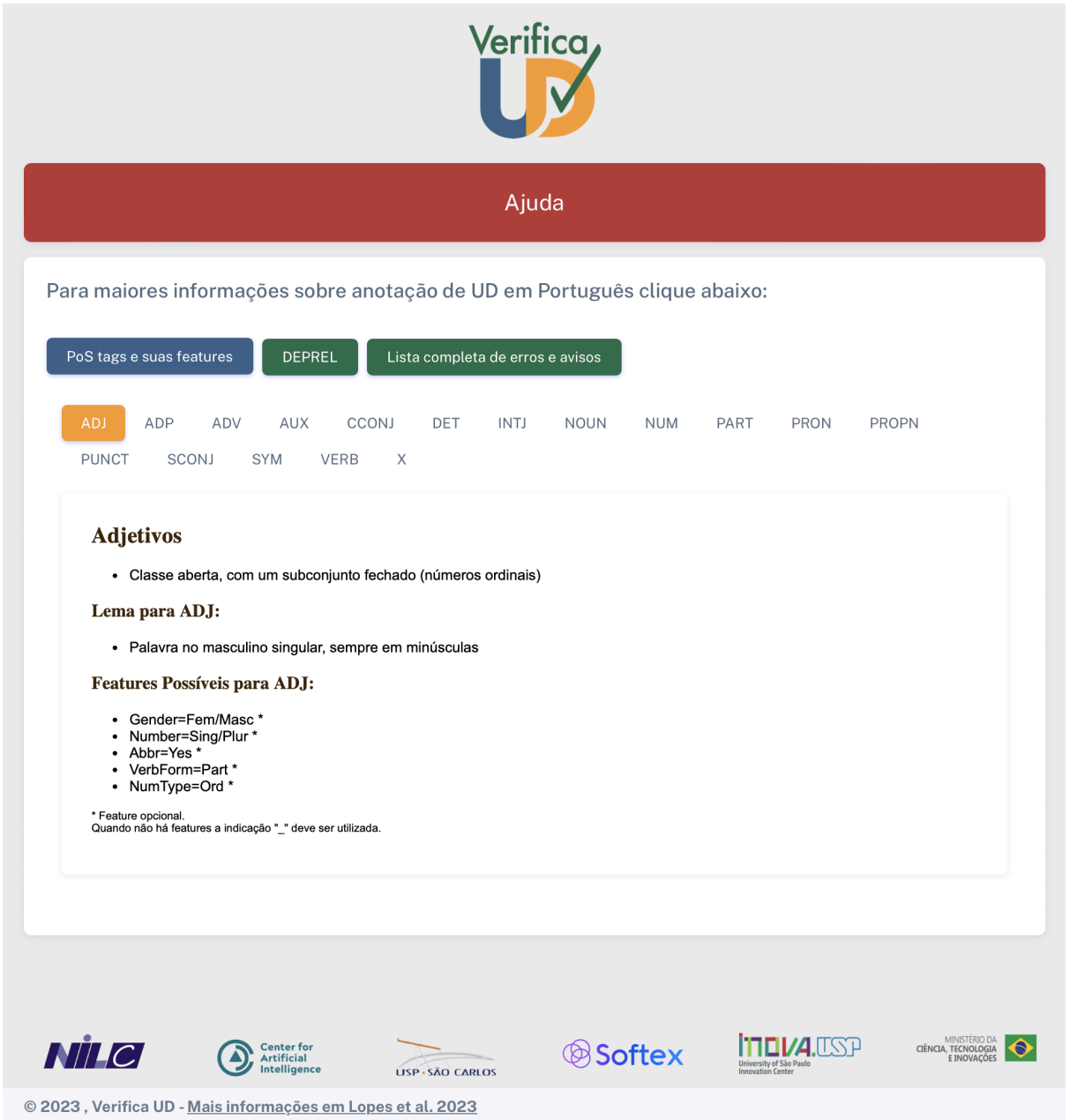


Figura 6. Tela do Verifica-UD apresentando a ajuda para o processo de Tagging.

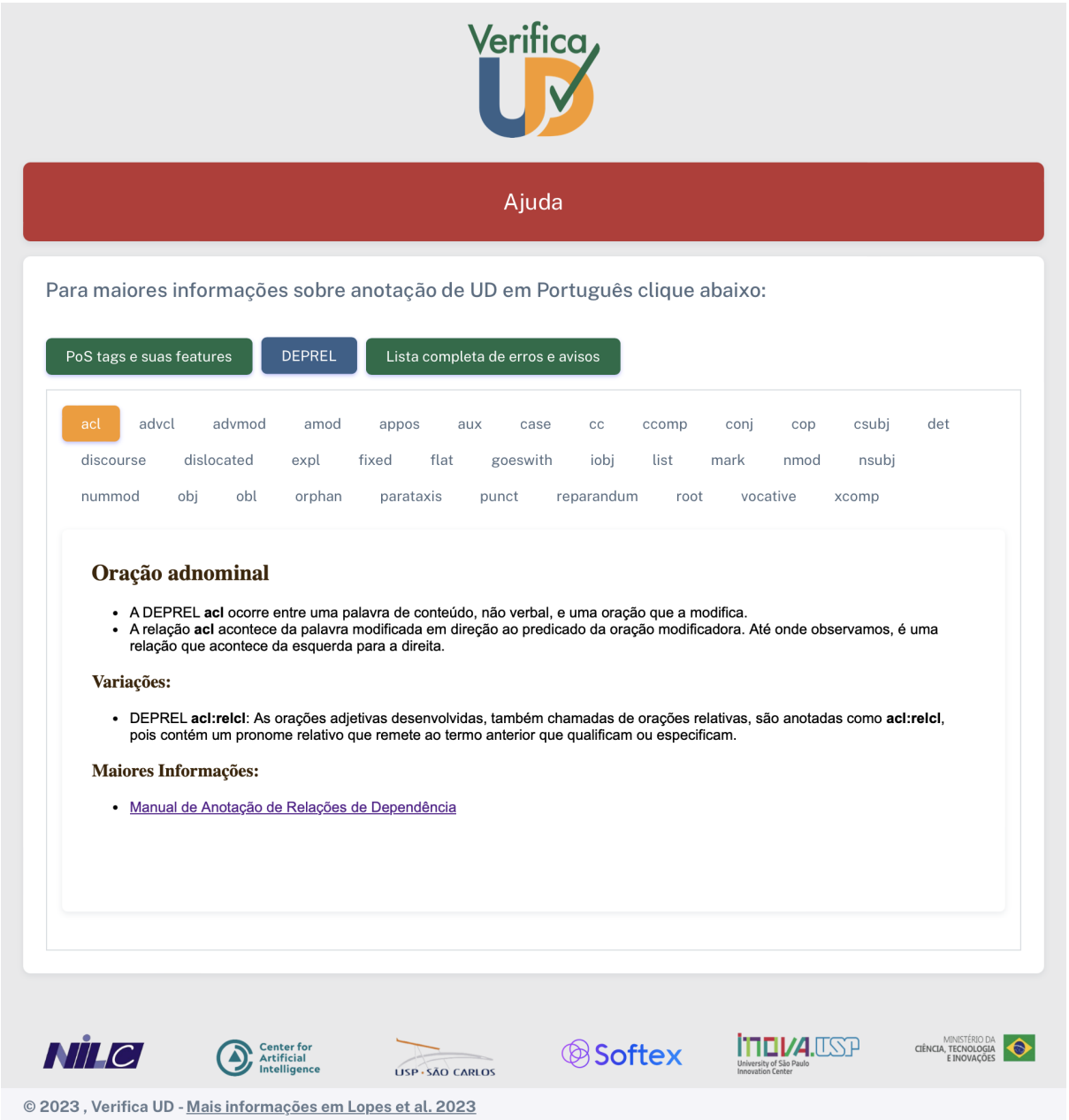


Figura 7. Tela do Verifica-UD apresentando a ajuda para o processo de Parsing.

Além disso, o usuário pode salvar localmente uma cópia de um relatório completo dos erros e avisos gerados pelo sistema por meio do botão "Salvar Relatório", que pode ser visto na Figura 4. Esse relatório é um arquivo texto que sumariza o resultado da verificação e lista os erros e avisos detectados, como pode ser visto na Figura 8. O exemplo apresentado corresponde à mesma execução descrita nas Figuras 4 e 5, e no relatório pode-se ver, por exemplo, os três erros e o aviso detectados para a sentença FOLHA_DOC009367_SENT024 nas linhas 18 a 21 do relatório.

```
Report Generated by Verifica-UD

File Name: corpus.conllu 7 sentenças (63 tokens)

Sentenças sem erros:           2
Sentenças sem erros ou avisos: 1
Nível Estrutural com: 1 erros (0 avisos)
Nível de Tagger com:  5 erros (1 avisos)
Nível de Parser com:  5 erros (0 avisos)

FOLHA_DOC000001_SENT003 token 1 : ERRO T21: Todo token PRON deve ter features 'PronType=Dem/Ind/Rel/Int/Prs' e de acordo com o tipo pode ter apenas 'Case=Acc/Dat/Nom', 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Person=1/2/3' e 'Poss=Yes'.
FOLHA_DOC000001_SENT003 token 12 : ERRO P37: Não deve haver Deprel projetivas (cruzamento de arcos) para nenhum token.
FOLHA_DOC000001_SENT003 token 12 : ERRO T24: Todo token PUNCT não deve ter features.
FOLHA_DOC000001_SENT003 token 8 : ERRO T17: Todo token NOUN pode ter apenas features 'Gender=Fem/Masc', 'Number=Sing/Plur' e 'Abbr=Yes'.
FOLHA_DOC000001_SENT003 token 9 : ERRO P37: Não deve haver Deprel projetivas (cruzamento de arcos) para nenhum token.
FOLHA_DOC000097_SENT056 token 6 : ERRO T12: Nenhum token ADV não abreviado pode ter features.
FOLHA_DOC000132_SENT031 token 7 : ERRO P08: Nenhum DET é head de relação - Exceções: pode ser head da relação fixed (e se for de fixed, pode ser de punct, cc e conj também), pode ser head de conj se o dependente for DET, pode ser head de cc se o token DET for dependente de conj.
FOLHA_DOC009367_SENT024 token 1 : ERRO P01: Todo token dependente de det é DET (o inverso não é verdadeiro).
FOLHA_DOC009367_SENT024 token 1 : ERRO T21: Todo token PRON deve ter features 'PronType=Dem/Ind/Rel/Int/Prs' e de acordo com o tipo pode ter apenas 'Case=Acc/Dat/Nom', 'Gender=Fem/Masc', 'Number=Sing/Plur', 'Person=1/2/3' e 'Poss=Yes'.
FOLHA_DOC009367_SENT024 token 4 : ERRO P57: Todo token head de aux:pass deve ter feature Voice=Pass.
FOLHA_DOC009367_SENT024 token 6 : AVISO p66: Normalmente, todo token PUNCT final tem como head o token root.
FOLHA_DOC031010_SENT007 token 3-4 : ERRO S02: Numeração errada de tokens concatenados (números errados).

```

Figura 8. Relatório gerado pelo Verifica-UD para a sentença descrita na Figura 5.

6. Considerações Finais

Neste relatório, apresentamos o Verifica-UD, uma ferramenta online para verificar possíveis erros em sentenças de córpus de português anotadas segundo o padrão UD. O propósito dessa ferramenta é auxiliar nos esforços de anotação para o português e, portanto, difundir em maior escala a anotação via UD por parte da comunidade brasileira de Processamento de Linguagem Natural.

A ferramenta online Verifica-UD na sua versão inicial já está disponível para uso nos endereços http://verificaud.icmc.usp.br e http://verificaud.icmc.usp.br:24080/verificaud/. Todas as informações referentes ao projeto POeTiSA podem ser encontradas no portal web em https://sites.google.com/icmc.usp.br/poetisa.

Novas versões da ferramenta poderão ser desenvolvidas no futuro com a adição de novas funcionalidades. Dentre elas, podemos citar:

- Entrada textual de sentenças via interface;
- Visualização de erros no texto do arquivo CoNLL-U;
- Edição dirigida do arquivo CoNLL-U.

Finalmente, esperamos que outras extensões e funcionalidades possam ser concebidas com base nos *feedbacks* dos usuários durante a utilização dessa primeira versão do Verifica-UD.

Agradecimentos

Este trabalho foi realizado no âmbito do Centro de Inteligência Articial da USP (C4AI - http://c4ai.inova.usp.br/), com o apoio da IBM e da FAPESP (processo 2019/07665-4). Este projeto também foi apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-Softex, coordenado pela Softex e publicado como Residência em TIC 13, DOU 01245.010222/2022-44.

Referências

de Marneffe, M.-C.; Manning, C.D.; Nivre, J.; Zeman, D. (2021). Universal Dependencies. Computational Linguistics, Vol. 47, N. 2, pages 255-308.

Duran, M.S. (2021). Manual de anotação de PoS tags: Orientações para anotação de etiquetas morfossintáticas em língua portuguesa, seguindo as diretrizes da abordagem Universal Dependencies (UD). Technical Report 434, ICMC-USP.

Duran, M.S. (2022). Manual de Anotação de Relações de Dependência - Versão Revisada e Estendida: Orientações para anotação de relações de dependência sintática em Língua Portuguesa, seguindo as diretrizes da abordagem Universal Dependencies (UD). Technical Report 440, ICMC-USP.

Duran, M.S.; Nunes, M.G.V.; Lopes, L.; Pardo, T.A.S. (2022). Manual de anotação como recurso de Processamento de Linguagem Natural: o modelo Universal Dependencies em língua portuguesa. Domínios de Lingu@gem, Vol. 16, N. 4, pages 1608-1643.

Duran, M.S.; Lopes, L.; Nunes, M.G.V.; Pardo, T.A.S. (2023). The Dawn of the Porttinari Multigenre Treebank: Introducing its Journalistic Portion. In the Proceedings of the 14th Symposium in Information and Human Language Technology (STIL), Belo Horizonte, Brazil. Em publicação.

Lopes, L.; Duran, M.S.; Fernandes, P.; Pardo, T.A.S. (2022). PortiLexicon-UD: a Portuguese lexical resource according to Universal Dependencies model. In the Proceedings of the Thirteenth Language Resources and Evaluation Conference, pages 6635-6643, Marseille, France.

Lopes, L.; Duran, M.S.; Pardo, T.A.S. (2023). Verifica-UD: a Verifier for Universal Dependencies Annotation for Portuguese. In the Proceedings of the 2nd Edition of the Universal Dependencies Brazilian Festival (UDFest-BR), Belo Horizonte, Brazil. Em publicação.

Nivre, J.; de Marneffe, M.-C.; Ginter, F.; Hajic, J.; Manning, C. D.; Pyysalo, S.; Schuster, S.; Tyers, F.; Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. In the Proceedings of the 12th Language Resources and Evaluation Conference, pages 4034-4043, Marseille, France.

Pardo, T.A.S.; Duran, M.S.; Lopes, L.; Di Felippo, A.; Roman, N.T.; Nunes, M.G.V. (2021). Porttinari - a large multi-genre treebank for brazilian portuguese. In the Proceedings of the XIII Symposium in Information and Human Language (STIL), pages 1-10.

Richardson, L.; Ruby, S. (2007). RESTful Web Services. O'Reilly, Beijing.