Utilizando um dicionário morfológico para expandir a cobertura lexical de uma gramática do português no formalismo HPSG

Ana Luiza Nunes[1], **Alexandre Rademaker**[1,3], **Leonel Figueiredo de Alencar**[1,2]

[1]Escola de Matemática Aplicada da FGV (FGV/EMAp), Brasil

[2]Universidade Federal do Ceará (UFC), Brasil

[3]IBM Research, Brasil

{analuizanunes8,arademaker}@gmail.com, leonel.de.alencar@ufc.br

Abstract. *The broad lexical coverage is one of the prerequisites for the robustness of computational grammar. We propose a methodology to populate the irregular verb forms of PorGram (a Portuguese grammar in the HPSG formalism) using data from MorphoBr. We implemented an algorithm that classifies the verb forms of MorphoBr into regular and irregular, applying the inflectional rules of PorGram. We evaluated the algorithm based on a sample of 38 verbs, both regular and irregular, obtaining the expected results. An additional contribution of the work was the improvement of MorphoBr, with the elimination of more than 270,000 wrong entries and the addition of almost 13,000 missing entries.*

Resumo. *Ampla cobertura lexical constitui um dos pré-requisitos para a robustez de uma gramática computacional. Propomos uma metodologia para povoar a tabela de formas verbais irregulares da PorGram (gramática do português no formalismo HPSG) utilizando os dados do dicionário MorphoBr. Nós implementamos um algoritmo que classifica as formas verbais do MorphoBr em regulares e irregulares, aplicando as regras flexionais da PorGram. Avaliamos o algoritmo com numa amostra de 38 verbos, regulares e irregulares, obtendo os resultados esperados. Uma contribuição adicional foi o melhoramento do MorphoBr, com a eliminação de mais de 270.000 entradas erradas e a inclusão de quase 13.000 entradas faltantes.*

1. Introdução

O processamento de linguagem natural enfrenta diversos desafios atrelados à dificuldade de cobertura dos fenômenos da linguagem. A modelagem direta de regras gramaticais e entradas lexicais em um formalismo computacional tem-se revelado uma solução válida em muitos contextos que exigem uma processamento sintático profundo, não obstante os inegáveis avanços das abordagens estatísticas baseadas em dados. Um exemplo prototípico é o sistema de resolução de perguntas Watson, da IBM, que integra em sua arquitetura híbrida uma gramática do inglês baseada na modelagem do conhecimento linguístico [Ferrucci et al. 2010, McCord et al. 2012].

O formalismo HPSG (*Head-driven Phrase Structure Grammar*) [Pollard 1994, Sag et al. 2003] é um dos mais difundidos para elaboração de gramáticas computacionais de ampla cobertura. O português dispõe de uma gramática nesse formalismo, que

é a LxGram [Branco 2014, Costa and Branco 2010]. Essa gramática, porém, não é distribuída sob licença de software livre e de código aberto (doravante FOSS, do inglês *free open-source software*).

Visando a preencher essa lacuna no terreno do *parsing* do português baseado em HPSG, foi iniciado recentemente o desenvolvimento da PorGram.[1] No momento, essa gramática modela um grande número de regularidades na conjugação verbal do português por meio de 130 regras lexicais, totalizando 395 regras de reescrita. Essas regras cobrem tanto os paradigmas tradicionalmente considerados regulares quanto casos de discordância gráfica (por exemplo, *venço*, *tanjo*, *ergo*, primeira pessoa do singular do presente do indicativo de *vencer*, *tanger* e *erguer*) e alternância vocálica (*sirvo* e *durmo*, formas de *servir* e *dormir*), entre outras variações sistemáticas do padrão geral [Cunha et al. 1985]. No entanto, um grande número de verbos, como *ter*, *dar*, *querer* etc., possui diversas formas completamente idiossincráticas, não abrangidas por essas regras.

Desse modo, o objetivo deste artigo é contribuir para a construção da PorGram através do preenchimento da tabela de formas irregulares, um dos componentes de uma gramática típica no sistema LKB, o ambiente de desenvolvimento utilizado na construção da PorGram [Copestake 2002]. Para esse preenchimento serão usadas as entradas do dicionário eletrônico MorphoBr [de Alencar et al. 2018]. Uma contribuição adicional deste trabalho é a melhoria do próprio MorphoBr, uma vez que, como veremos, o algoritmo de classificação de formas verbais permitiu identificar e corrigir dezenas de milhares de erros nesse recurso.

As próximas seções apresentam o recurso lexical MorphoBr e a gramática PorGram, em seguida são descritos os passos para o preenchimento da tabela de formas irregulares. Por fim, apresentamos os resultados alcançados e as conclusões.

2. MorphoBr

O MorphoBr [de Alencar et al. 2018] é um léxico de formas plenas de alta cobertura. Abrange as formas flexionadas de substantivos, adjetivos e verbos, que são as classes gramaticais mais numerosas do português. Foi criado a partir da combinação, revisão e expansão dos dicionários eletrônicos Label-Lex [Eleutério et al. 1995] e DELAF-PB [Muniz 2004]. Incorpora, também, os neologismos gerados por [Silva 2019] por meio da aplicação de regras produtivas de formação de palavras.

Para as classes flexionáveis, as entradas do MorphoBr constituem-se de pares de forma flexionada e lema seguido de informações morfológicas, conforme a *Listing* 1. Adjetivos e substantivos totalizam mais de meio milhão de entradas cada, verbos somam mais de dois e meio milhões (não contabilizando as formas com pronomes clíticos), correspondentes a 28080 lemas.[2]

3. PorGram

Numa primeira fase, a exemplo de gramáticas análogas, como a LxGram, a PorGram está sendo implementada com o sistema *Grammar Matrix*, que permite gerar o código de uma

[1] A motivação e a estrutura dessa gramática são descritas em detalhe em artigo, de autoria do segundo e terceiro autor do presente trabalho, submetido recentemente a um periódico, ora em revisão por pares.

[2] Dados computados em versão do MorphoBr anterior às modificaçòes descritas no presente artigo.

Listing 1. Exemplos de entradas do MorphoBr

```
1 venço   vencer+V+PRS+1+SG
2 ergo    erguer+V+PRS+1+SG
3 ajo agir+V+PRS+1+SG
4 durmo   dormir+V+PRS+1+SG
5 tenho   ter+V+PRS+1+SG
6 dou     dar+V+PRS+1+SG
```

Listing 2. Regra da terceira pessoa do plural do presente do indicativo

```
1 pres-ind-3pl-suffix :=
2 ; (partir partem) (vender vendem) (doar doam) (passear
     ↪ passeiam)
3 %suffix (ir em) (er em) (!zar !zam) (ear eiam)
4 pres-ind-3pl-lex-rule.
```

gramática inicial a partir da descrição de propriedades gramaticais da língua por meio do preenchimento de um questionário de customização [Bender et al. 2010]. Para testar a gramática, tem sido utilizado o sistema LKB, implementado em LISP [Copestake 2002].

Uma das limitações conhecidas da *Grammar Matrix* é que, no terreno da morfologia, se limita à morfotática [Goodman 2013], não permitindo modelar regras de alternância morfofonológica ou ortográfica [Beesley and Karttunen 2003]. Por exemplo, no questionário da *Grammar Matrix*, podemos modelar a formação da primeira pessoa do singular do presente do indicativo por meio da adjunção do sufixo *o* ao radical verbal. Essa regra funciona para um verbo como *comprar*. No entanto, não contempla nenhuma das formas da *Listing* 1. Enquanto as duas últimas formas discrepam idiossincraticamente do padrão conjugacional regular, as quatro primeiras exemplificam variações sistemáticas no radical verbal ou na flexão que afetam milhares de formas verbais. Desse modo, as regras codificadas inicialmente usando a *Grammar Matrix* foram modificadas manualmente, de modo a incluir o maior número possível de padrões de flexão e, assim, diminuir a quantidade de memória ocupada pela gramática.

A PorGram é codificada na linguagem *Type Description Language* (TDL), que possibilita a declaração de entradas lexicais, regras sintagmáticas e regras lexicais, como na *Listing* 2, que constitui recodificação manual de regra gerada inicialmente pela *Grammar Matrix*.

A segunda linha da *Listing* 2 é um comentário com exemplos. Na terceira linha, temos uma sequência de regras de reescrita de sufixos, que são processados sucessivamente pelo algoritmo de *parsing* do LKB [Copestake 2002, Goodman 2013]. Em cada uma dessas regras de rescrita, o primeiro sufixo constitui a entrada, enquanto o segundo constitui a saída da regra. Por exemplo, se o lema verbal termina em *ear*, a flexão deve ser *eiam*. No par de sufixos `(!zar !zam)`, `!z` é uma classe de caracteres equivalente *grosso modo* à expressão regular `[^e]`. Ou seja, *ar* só é substituído por *am* se o caractere precedente não for *e*, o que evita a hipergeração de formas agramaticais como *passeam*.

As formas irregulares, que não são geradas através das regras flexionais, precisam

Listing 3. Exemplos de entradas de formas irregulares

```
1 têm PRES-IND-3PL-SUFFIX ter
2 vão PRES-IND-3PL-SUFFIX ir
3 extinguido PAST-PART-SUFFIX extinguir
4 extinto PAST-PART-SUFFIX extinguir
```

ser listadas em uma tabela, como exemplificado na *Listing* 3. Na PorGram, essa tabela é armazenada no arquivo `my-irregs.tab`. Observe que, quando uma forma irregular coexiste com uma regular, como no caso do particípio passado de *extinguir*, ambas precisam ser listadas nesse arquivo.[3]

4. Metodologia de preenchimento da tabela de formas irregulares

Com o intuito de povoar `my-irregs.tab` a partir das entradas do MorphoBr, implementamos um algoritmo que classifica as formas verbais em regulares ou irregulares, com base nas regras flexionais da PorGram. A linguagem de programação escolhida para essa tarefa foi Haskell, que é puramente funcional e tem como vantagens a clareza e simplicidade do código.

Na PorGram, as declarações de regras lexicais, como exemplificado na *Listing* 2, são formuladas no arquivo `my-irules.tdl`. Infelizmente ainda não existe um *parser* para a leitura de arquivos TDL em Haskell. Por isso, utilizamos a biblioteca PyDelphin[4], que possibilita armazenar, em formato JSON, os atributos dos objetos `LetterSet`, que modela as classes de caracteres, e `LexicalRuleDefinition`, que modela as regras flexionais.

Dadas as diferenças de formato de representação do MorphoBr e da PorGram, criamos manualmente um arquivo contendo as etiquetas correspondentes a cada regra da gramática. Em Haskell, a partir do arquivo JSON com os atributos dos objetos `LetterSet` e `LexicalRuleDefinition`, os padrões das regras de reescrita foram codificados como expressões regulares. Devido ao grande volume de entradas verbais do MorphoBr, utilizamos uma estrutura eficiente para organizá-las, o `Data.Map`. Nele, como em um dicionário, temos chaves buscáveis e valores associados às mesmas. O `Map` das entradas tem como chave os lemas, aos quais se associam listas de tuplas em que o primeiro elemento é a forma flexionada e o segundo é a regra correspondente ao conjunto de etiquetas, quando não há uma regra correspondente o segundo elemento é uma *string* vazia.

Para obter as formas flexionadas irregulares, o algoritmo gera, para cada forma do MorphoBr contemplada por uma regra da gramática, uma nova forma através da aplicação dessa regra no lema. Por exemplo, a aplicação da regra lexical PAST-PART-SUFFIX a *extinguir*, que forma o particípio passado, produz, pelo padrão `(guir guido)`, a forma regular *extinguido*. Ao analisar a forma *extinto*, o algoritmo a compara com a que produziu, ou seja, *extinguido*, por meio da função `isRegular`

[3]Para mais detalhes sobre o funcionamento do algoritmo de *parsing* e geração morfológicos do LKB, ver [Copestake 2002].

[4]`https://github.com/delph-in/pydelphin`

e, comohttps://www.overleaf.com/project/60ef62474d10575be26f47d5 essa forma é diferente, a adiciona à tabela `my-irregs.tab`. Como vimos, nesse caso, tanto a forma irregular *extinto* quanto a regular *extinguido* são adicionadas à tabela.

A função `isRegular` tem três possíveis retornos:

1. no caso em que a forma analisada coincide com a forma produzida pela aplicação da regra, ou seja, é regular, o retorno é vazio;
2. quando a forma analisada não é igual à forma produzida pela regra, porém a forma produzida existe no MorphoBr, as duas formas são retornadas;
3. se as formas são diferentes e a forma que foi produzida pela regra não existe no MorphoBr, retorna apenas a forma analisada.

5. Resultados

O resultado esperado do algoritmo de classificação delineado acima era uma tabela de exceções com aproximadamente 10 mil entradas. Esse número era uma mera estimativa baseada na *English Resource Grammar* (ERG), aparentemente, a maior gramática implementada no formalismo HPSG [Flickinger 2000]. Nessa gramática, a tabela correspondente possui 4184 formas de verbos, correspondentes a 808 lemas verbais, ao passo que o léxico principal dessa gramática contém 4346 lemas verbais diferentes.[5]

Contrariando largamente à nossa expectativa inicial, ao executarmos o algoritmo sobre as formas verbais do MorphoBr pela primeira vez, obtivemos uma tabela com 483683 entradas. Uma análise cuidadosa desses dados indicou que o grande volume não resultava de erros de classificação do algoritmo nem de modelagem inadequada das regras flexionais. Em vez disso, foi causado por formas espúrias existentes no MorphoBr, que se subdividiam em 5 tipos principais:

1. formas no infinitivo terminadas em *á*, *ê*, *i*, *í* ou *ô*;
2. formas não imperativas na primeira ou segunda pessoa do plural não terminadas em *s*;
3. formas na segunda pessoa do singular que não do imperativo ou presente do indicativo sem terminar em *s*;
4. formas com o sufixo *ásseis* em vez de *asseis*;
5. formas com erros diversos, como *veiste* em vez de *vieste* ou *curguei* em vez de *curvei*.

A maior parte dessas formas espúrias eram duplicatas de formas corretas. Por meio de regras baseadas nesses tipos, foram eliminadas 270278 formas espúrias e incluídas 12908 entradas corrigidas. Essas correções permitiram reduzir a tabela, em nova aplicação do algoritmo de classificação, a 11581 entradas.

Uma primeira análise das entradas dessa tabela aponta para a necessidade de mais correções nas formas verbais do MorphoBr. Como podemos constatar na *Listing* 4, a segunda variante de cada uma dessas formas ou está com a ortografia desatualizada ou constitui uma forma espúria. Pelo Acordo Ortográfico da Língua Portuguesa de 2009, as formas *abóia*, *abotôo* e *adeqüemos* não se escrevem mais com diacríticos. A forma

[5] A ERG tem sido continuamente desenvolvida desde o seu lançamento há mais de duas décadas, sendo extremamente complexa. Não excluímos o fato de que possua mais entradas de verbos em bancos de dados auxiliares.

Listing 4. Exemplos da tabela final de formas irregulares

```
1  aboia    PRES-IND-3SG-SUFFIX     aboiar
2  abóia    PRES-IND-3SG-SUFFIX     aboiar
3  abotoo   PRES-IND-1SG-SUFFIX     abotoar
4  abotôo   PRES-IND-1SG-SUFFIX     abotoar
5  abstido  PAST-PART-SUFFIX        abster
6  absteido         PAST-PART-SUFFIX        abster
7  adequemos        PRES-SUBJ-1PL-SUFFIX    adequar
8  adeqüemos        PRES-SUBJ-1PL-SUFFIX    adequar
9  dávamos  IMPF-IND-1PL-SUFFIX     dar
10 demos    IMPF-IND-1PL-SUFFIX     dar
11 quises   FUT-SUBJ-2SG-SUFFIX     querer
12 quiseres         FUT-SUBJ-2SG-SUFFIX     querer
13 reavemos         PRES-IND-1PL-SUFFIX     reaver
14 reemos   PRES-IND-1PL-SUFFIX     reaver
```

demos não constitui forma do imperfeito do indicativo de *dar*, sendo *dávamos* a única forma correta para a primeira pessoal do plural nesse caso. Finalmente, as formas *quises* e *reemos* não integram os paradigmas de *querer* e *reaver*, respectivamente. As únicas formas corretas para as combinações de pessoa, número, tempo e modo indicados na *Listing* 4 são *quiseres* e *reavemos*.

Além de melhorias no MorphoBr, a produção da tabela também possibilitou uma revisão da cobertura das regras definidas em `my-irules.tdl`, chamando a atenção para a ausência do padrão `(er ias)` na regra `fut-pret-2sg-suffix`, que foi atualizada.

Como uma primeira forma de avaliação do algoritmo, aplicamo-lo a uma amostra de verbos altamente irregulares, constituída das entradas do MorphoBr para os verbos *dar*, *dizer*, *estar*, *fazer*, *haver*, *ir*, *poder*, *querer*, *saber*, *ser*, *ter*, *trazer*, *ver* e *vir*. Apesar de irregulares na maior parte da conjugação, esses verbos possuem algumas formas regulares. O algoritmo gerou uma tabela com as classificações esperadas para essa amostra de verbos. Por exemplo, no caso do verbo *dar*, a tabela não contém formas como *damos*, *dava* ou *daríamos*, que são geradas pelas regras flexionais e não possuem variantes irregulares no MorphoBr, apenas formas idiossincráticas como *dou*, *deu*, *déssemos* ou *déramos*, não abrangidas por essas regras.

Por outro lado, verificamos que a tabela não inclui nenhuma forma de uma amostra de verbos totalmente regulares, ou seja, cuja conjugação segue rigorosamente as regras flexionais implementadas, com exceção de variantes com ortografia anterior ao Acordo Ortográfico ou que constituem formas espúrias, como exemplificamos na *Listing* 4. Essa segunda amostra consistiu dos verbos *acuar*, *advertir*, *agir*, *atacar*, *caçar*, *chegar*, *comprar*, *distinguir*, *doar*, *dormir*, *erguer*, *ferir*, *mentir*, *partir*, *passear*, *perseguir*, *proteger*, *puir*, *ressarcir*, *seguir*, *sentir*, *vencer*, *vender* e *vestir*. Foi extraída dos comentários que documentam as diferentes regras flexionais do arquivo `my-irules.tdl`.

6. Conclusão

Neste artigo, relatamos sobre um algoritmo em Haskell para o preenchimento da tabela de formas verbais irregulares da gramática PorGram, a fim de que todo o léxico verbal do MorphoBr possa vir a ser utilizado pela gramática. Antes disso, a gramática só era capaz de analisar as formas regulares contempladas pelas regras flexionais. A primeira aplicação desse algoritmo nos mais de dois e meio milhões de formas verbais do MorphoBr resultou em uma tabela com quase meio milhão de formas, contrariando amplamente a expectativa inicial de algo em torno de 10 mil entradas.

Um exame da tabela gerada permitiu, por um lado, identificar um grupo de cinco tipos de erros no MorphoBr, levando a uma correção do recurso que eliminou 270278 formas espúrias e inseriu 12908 entradas corrigidas. Por outro lado, possibilitou constatar e corrigir um erro numa das regras flexionais. Aplicado sobre os dados atualizados tanto do MorphoBr quanto da gramática, o algoritmo de classificação produziu uma tabela com apenas 11581 entradas.

Uma avaliação manual dessa tabela constituiria um processo demorado. Por isso, optamos por uma avaliação por amostragem. Por um lado, analisamos os resultados para uma amostra de 14 verbos altamente irregulares. A tabela apresentou, nesse caso, as classificações esperadas. Por outro lado, a tabela não inclui formas indevidas de verbos regulares, a julgar pela análise dos resultados para 24 desses verbos, que instanciam os diferentes padrões de regularidade modelados pela PorGram. Desse modo, a nossa expectativa é de que a gramática agora possa analisar qualquer forma verbal do MorphoBr. Para tanto, temos ainda de corrigir as formas verbais erradas remanescentes do recurso e elaborar uma metodologia eficiente de avaliação em larga escala, dado o grande volume de dados envolvido.

Outro trabalho a ser desenvolvido num futuro próximo é revisar as regras flexionais, de modo a possivelmente incluir mais padrões regulares, e testar essas regras sob a perspectiva não apenas da análise, como fizemos aqui, mas também da geração, utilizando a capacidade geradora do algoritmo de classificação.

Referências

Beesley, K. R. and Karttunen, L. (2003). *Finite state morphology*. CSLI, Stanford, California.

Bender, E. M., Drellishak, S., Fokkens, A., Poulson, L., and Saleem, S. (2010). Grammar customization. *Research on Language & Computation*, 8(1):23–72. 10.1007/s11168-010-9070-1.

Branco, A. e. F. C. (2014). A computational grammar for deep linguistic processing of Portuguese: LXGram (version 5). Technical report, Universidade de Lisboa, Departamento de Informática.

Copestake, A. (2002). *Implementing typed feature structure grammars*. CSLI, Stanford, California.

Costa, F. and Branco, A. (2010). LXGram: A deep linguistic processing grammar for Portuguese. In Pardo, T. A. S., Branco, A., Klautau, A., Vieira, R., and de Lima, V. L. S., editors, *Computational Processing of the Portuguese Language*, pages 86–89, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cunha, C., Cintra, L. F. L., et al. (1985). *Nova gramática do português contemporâneo*. Nova Fronteira Rio de Janeiro.

de Alencar, L. F., Cuconato, B., and Rademaker, A. (2018). MorphoBr: An open source large-coverage full-form lexicon for morphological analysis of Portuguese. *Texto Livre: Linguagem e Tecnologia*, 11(3):1–25.

Eleutério, S., Freire, H., Ranchhod, E., and Baptista, J. (1995). A system of electronic dictionaries of Portuguese. *Lingvisticae Investigationes*, 19(1):57–82.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79.

Flickinger, D. (2000). On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Goodman, M. W. (2013). Generation of machine-readable morphological rules with human readable input. *University of Washington Working Papers in Linguistics*, 30:1–34.

McCord, M. C., Murdock, J. W., and Boguraev, B. K. (2012). Deep parsing in Watson. *IBM Journal of Research and Development*, 56(3.4):3:1–3:15.

Muniz, M. C. M. (2004). A construção de recursos linguístico-computacionais para o português do brasil: o projeto de unitex-pb. *Master's thesis, Instituto de Ciências Matemáticas e de Computação, USP*.

Pollard, C. (1994). *Head-driven Phrase Structure Grammar*. CSLI, Chicago, Illinois.

Sag, I. A., Wasow, T., and Bender, E. M. (2003). *Syntactic theory: A formal introduction*. University of Chicago Press, Chicago, second edition edition.

Silva, H. L. B. (2019). Expansão do MorphoBr através da modelagem computacional de processos de formação de palavras em português. Master's thesis, Universidade Federal do Ceará, Brazil.