Pipeline para identificação de erros lexicais e geração de sugestões de correção

Luana Q. Garcia[1], **Miguel H. Chinellato**[1],
Helena de M. Caseli[1], **Leandro H. M. Oliveira**[2]

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil

[2]Empresa Brasileira de Pesquisa Agropecuária – Embrapa

{luanaqg,miguel.chinellato}@estudante.ufscar.br,

helenacaseli@ufscar.br, leandro.oliveira@embrapa.br

Resumo. *No PLN, os textos são a principal fonte de informação na geração de modelos computacionais usando aprendizado de máquina. Entretanto, para que sejam úteis no processo de aprendizado, estes textos precisam representar corretamente o fenômeno que se deseja aprender e, neste caso, os erros lexicais podem ser impactantes. Este artigo apresenta a proposta de um pipeline para preparação e/ou correção de textos que identifica várias categorias de erros lexicais. O pipeline objetiva identificar, anotar e categorizar os erros contidos nos textos, bem como sugerir correções de forma automática.*

1. Introdução

Na atualidade, a geração de modelos computacionais a partir de textos é realizada principalmente via Aprendizado de Máquina (AM). No *pipeline* de AM, a geração desses modelos passa por várias etapas das quais a de preparação de dados é a que demanda mais tempo [Chu et al. 2016], uma vez que modelos de alta performance exigem dados de qualidade para que se possa aprender padrões de alta qualidade [Zhang et al. 2003]. Embora a limpeza, correção e conversão dos dados sejam tarefas comuns e interdependentes [Ilyas e Rekatsinas 2022] em AM com dados textuais, o mesmo não se pode dizer sobre a descrição detalhada e ordenada dos passos realizados [Parulian e Ludäscher 2023].

Assim, tendo como referência algumas ferramentas de limpeza de dados para dados tabulares [Parulian e Ludäscher 2023, Li et al. 2023], a principal contribuição deste trabalho está na definição de um *pipeline* de preparação e/ou correção de textos (dados não estruturados) que pode ser aplicado a diversos domínios, operacionalizando esta tarefa e diminuindo débito técnico [Sculley et al. 2015] a longo prazo no modelo. Este artigo descreve a instanciação do *pipeline* para o domínio da agropecuária. O código do *pipeline* está disponível livremente[1].

2. O *pipeline* de correção textual

O *pipeline* de correção textual proposto neste trabalho identifica, anota e categoriza, de forma automática, os erros lexicais contidos nos textos. Para tanto, utiliza como recursos linguísticos: (1) um léxico do domínio geral composto por palavras do idioma sendo

[1]Disponível em: https://github.com/LALIC-UFSCar/pie-embrapa-pln

processado e (2) uma lista de *stopwords* nesse mesmo idioma. A partir desses recursos, inicialmente, os *types* (palavras únicas) presentes nos textos são identificados e todos os *types* (exceto *stopwords*) não encontrados no léxico de domínio geral são classificados como palavras *desconhecidas*.

A partir do conjunto de *desconhecidas*, o *pipeline* identifica e categoriza as palavras em: (i) falta de acento, (ii) acentuação incorreta, (iii) erro de grafia e (iv) palavras aglutinadas[2]. Sugestões de correção também são geradas. Outras categorias também são identificadas neste processo, tais quais: números, locais, termos específicos do domínio e palavras estrangeiras, mas estas não são consideradas erros.

Na identificação e categorização dos erros, o *pipeline* utiliza, além do léxico de domínio geral já mencionado, (3) uma base de dados de nomes de países, regiões, estados e municípios, (4) um léxico especializado do domínio, (5) um dicionário de frequências de palavras do idioma dos textos e (6) uma ferramenta de tradução para identificação de palavras em outras línguas. A Figura 1 ilustra os passos de categorização das palavras *desconhecidas* e seus respectivos recursos utilizados.

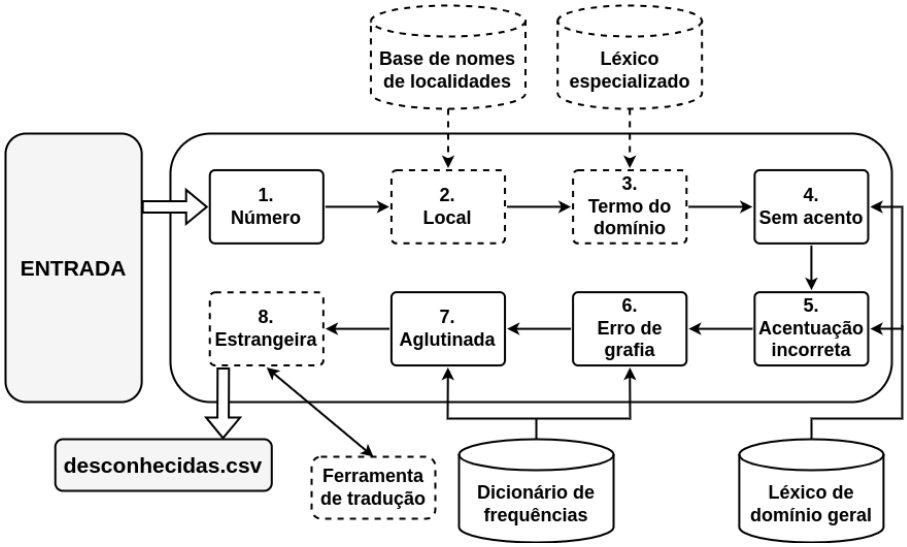

Figura 1. *Pipeline* de Categorização

A identificação e categorização dos erros ocorre na seguinte sequência de passos:

1. **verificação de números –** verifica se o *type* desconhecido é um número por meio do uso da função `isnumeric()` do Python, a linguagem de programação usada para implementação do *pipeline*;
2. **verificação de nomes de locais (opcional) –** identifica ocorrências de nomes de países, gentílicos, regiões, estados e municípios utilizando uma base de nomes de localidades;
3. **verificação de termos do domínio (opcional) –** verifica se a palavra desconhecida é um termo do domínio por meio da consulta a uma base de termos específicos da agropecuária, inseridos na língua portuguesa;

[2]Palavras que originalmente aparecem juntas no texto mas deveriam aparecer separadas.

4. **verificação de falta de acento –** verifica *types* que pertencem ao léxico de domínio geral da língua portuguesa, porém com todas suas palavras desacentuadas, identificando possíveis palavras sem acento;
5. **verificação de acentuação incorreta –** são removidos os acentos das próprias palavras *desconhecidas* para, em seguida, verificar se, agora, estas pertencem ao mesmo léxico desacentuado do passo anterior, identificando palavras com possíveis acentos errados;
6. **verificação de erros de grafia –** identifica erros de grafia por meio da biblioteca do algoritmo Symspell[3] para Python, que corrige palavras com pequenos erros de grafia através de distâncias de edição com base no dicionário de frequências de palavras do idioma desejado;
7. **verificação de palavras aglutinadas –** separa, com espaços em branco, palavras aglutinadas por meio de uma função[4] que também utiliza o dicionário de frequência de palavras e assume que essas frequências seguem a Lei de Zipf;
8. **verificação de palavras estrangeiras (opcional) –** verifica se as palavras *desconhecidas* são possíveis palavras ou termos em outras línguas, nomeadamente o inglês, espanhol e francês, por meio de uma ferramenta de tradução.

A saída do *pipeline* é um arquivo de extensão csv intitulado `desconhecidas.csv` contendo 3 colunas: **palavra** (palavra identificada como desconhecida), **sugestão** (sugestão de correção do *type* desconhecido, nas categorias que são consideradas erros léxicos) e **categoria** (etapa do *pipeline* em que a palavra foi categorizada, sendo *desconhecida* caso nenhuma das categorias tenha sido detectada).

2.1. Definição da ordem de execução do *pipeline*

Diversos experimentos foram realizados em um córpus do domínio agropecuário composto pelos textos dos projetos da Embrapa para identificar a sequência dos passos previamente descritos, resultando na ordem ilustrada na Figura 1. Com o intuito de obter melhores resultados, optou-se por inserir nos passos inicias do *pipeline* as verificações que se mostraram mais assertivas, notadamente: a verificação de números, de nomes de locais e dos termos específicos do domínio. Os próximos passos foram escolhidos de forma a, novamente, priorizar aqueles que demonstraram ser mais precisos durante testes empíricos. Desse modo, os passos seguintes são, respectivamente, as verificações de falta de acento e de acentuação incorreta. Ambos são bastante similares no quesito assertividade e funcionamento, utilizando o léxico desacentuado conforme descrito anteriormente.

Por fim, as últimas etapas do *pipeline* (antes da verificação de palavras estrangeiras) foram definidas como sendo, respectivamente, as verificações de erros de grafia e de palavras aglutinadas. Em ambos os casos, as funções retornam sugestões de correção corretas quando trata-se mesmo de um erro, com palavras como `princpais` e `substutuindo` sendo corrigidas, respectivamente, para `principais` e `substituindo` pelo Symspell; e termos aglutinados como `comênfase` sendo corrigidos para `com ênfase` pela função de separação de palavras. Contudo, tais passos foram deixados mais perto do fim do *pipeline* pois apresentavam, com frequência,

[3]Disponível em: `https://pypi.org/project/symspellpy/`. Acesso em: 30 jun. 2023.

[4]Implementação baseada em: `https://stackoverflow.com/a/11642687`. Acesso em: 30 jun. 2023.

sugestões de correção para palavras que não eram erradas (como os termos específicos do domínio agropecuário). Por exemplo, o Symspell sugeriu corrigir o termo `apomíticos` para `apolíticos`, assim como `nematófagos` para `hematófagos`, ambas palavras corretas porém particulares do domínio investigado. De maneira similar, a rotina de separação de *tokens* sugeriu corrigir `negligenciamento` para `negligencia mento`. Entretanto, muito mais numerosos eram os casos em que esta função de separação sugeria separar neologismos, como `nanopartículas` para `nano partículas`, `agroativos` para `agro ativos` e uma abundância de outros neologismos do contexto agropecuário. Escolheu-se deixar a verificação de palavras estrangeiras como o último do *pipeline* devido, sobretudo, a sua baixa assertividade.[5]

2.2. Instanciação do *pipeline* para o português do Brasil

Com intuito de exemplificar o uso do *pipeline* proposto, foi realizada a instanciação para processar textos escritos em português do Brasil no domínio da agropecuária contendo resumos de projetos da Embrapa. Para tanto, os recursos lexicais usados ao longo dos passos foram: **1. Léxico de domínio geral:** léxico POeTiSA (POrtuguese processing - Towards Syntactic Analysis and parsing)[6]; **2. Lista de *stopwords*:** *stopwords* da língua portuguesa do NLTK [Bird et al. 2009] [7]; **3. Base de nomes de localidades:** API do IBGE para Python[8] que cobre os nomes de quase todos os municípios, estados e regiões do Brasil; e um arquivo JSON[9] que contém os nomes e respectivos gentílicos de todos os países do Google Maps; **4. Léxico especializado:** AgroTermos[10], que contém um conjunto de termos e conceitos específicos inter-relacionados semanticamente pertencente ao domínio agropecuário; **5. Dicionário de frequências:** dicionário de frequências de palavras da língua portuguesa extraído de todos os corpora disponibilizados pela Linguateca[11]; **6. Ferramenta de tradução:** API do Google Tradutor para Python[12] objetivando a tradução de palavras da língua inglesa, espanhola e francesa.

3. Considerações finais

Tendo como foco a operacionalização do processamento de textos de um domínio específico, contendo possíveis termos especializados e erros de digitação/grafia produzidos por agentes humanos ou conversão automática de formato (por exemplo, de PDF para TXT), este artigo apresentou um *pipeline* de preparação e/ou correção de textos. Esse *pipeline* tem a grande vantagem de evidenciar os erros dos textos, separando o que é de fato erro e o que não é, além de oferecer sugestões para a correção humana.

[5]Nos experimentos realizados, notou-se que a API do Google Tradutor encontrava traduções errôneas para vários termos. Vale ainda ressaltar que esta é, com uma boa margem, o passo mais demorado de todo o *pipeline* uma vez que tal API funciona por meio de requisições Web, resultando em uma grande demora para retornar as traduções.

[6]Disponível em: `https://sites.google.com/icmc.usp.br/poetisa`. Acesso em: 30 jun. 2023.

[7]Disponível em: `https://www.nltk.org`. Acesso em: 30 jun. 2023.

[8]Disponível em: `https://pypi.org/project/ibge/`. Acesso em: 30 jun. 2023.

[9]Disponível em: `https://gist.github.com/jonasruth/61bde1fcf0893bd35eea`. Acesso em: 30 jun. 2023.

[10]Disponível em: `https://sistemas.sede.embrapa.br/agrotermos/`. Acesso em: 30 jun. 2023.

[11]Disponível em: `https://linguateca.pt/acesso/info_freq_English.php`. Acesso em: 30 jun. 2023.

[12]Disponível em: `https://pypi.org/project/googletrans/`. Acesso em: 30 jun. 2023.

Agradecimentos

Este trabalho é resultado do Projeto Integrador Extensionista (PIE) vinculado ao Projeto de Extensão da UFSCar "Geração de representações de palavras no domínio agropecuário para melhoria dos processos de gestão da informação e conhecimento da Embrapa" (23112.035676/2022-76). Esse projeto é uma parceria com a Embrapa no projeto "Infopasto: Gestão da Informação e do conhecimento como suporte à gestão estratégia do Portfólio de Pastagens da Embrapa" (40.18.03.058.00.00) financiado pelo CNPq, a quem agradecemos o suporte financeiro.

Referências

Bird, S., Klein, E., e Loper, E. (2009). *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly, 1st edition.

Chu, X., Ilyas, I. F., Krishnan, S., e Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 2201–2206, New York, NY, USA. Association for Computing Machinery.

Ilyas, I. F. e Rekatsinas, T. (2022). Machine learning and data cleaning: Which serves the other? *J. Data and Information Quality*, 14(3).

Li, P., Chen, Z., Chu, X., e Rong, K. (2023). Diffprep: Differentiable data preprocessing pipeline search for learning over tabular data. *Proc. ACM Manag. Data*, 1(2).

Parulian, N. N. e Ludäscher, B. (2023). Trust the process: Analyzing prospective provenance for data cleaning. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 1513–1523, New York, NY, USA. Association for Computing Machinery.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., e Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2503–2511, Cambridge, MA, USA. MIT Press.

Zhang, S., Zhang, C., e Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence*, 17:375–381.