

# ***Intelli3text: Uma Biblioteca Python para Ingestão, Limpeza e Normalização de Textos***

**Jefferson Rodrigo Speck<sup>1</sup>, Luiz Henrique Zavatini Feltrin<sup>1</sup>**

<sup>1</sup> Faculdade Donaduzzi, Toledo – PR – Brasil

`jefferson.speck@biopark.com.br, luizhzfeltrin@gmail.com`

**Abstract.** *This study presents intelli3text, a Python library for Natural Language Processing preprocessing. This work addresses the issues of fragmentation and noise during text ingestion from heterogeneous sources. Through a modular architecture, the library offers a unified pipeline for text cleaning, paragraph-level language identification, and normalization. The primary contribution lies in enhancing traceability and auditability within process workflows by providing evidence reports and dynamic management of language models.*

**Resumo.** *Este estudo apresenta a intelli3text, uma biblioteca Python para pré-processamento de Processamento de Linguagem Natural. O trabalho aborda o problema da fragmentação e do ruído na ingestão de textos de fontes heterogêneas. Através de uma arquitetura modular a biblioteca oferece um pipeline unificada para limpeza, identificação de idioma por parágrafo e normalização. A principal contribuição reside na ampliação da rastreabilidade e auditabilidade em fluxos de processamento através de relatórios de evidências e gerenciamento dinâmico de modelos linguísticos.*

## **1. Introdução e Problematização**

A performance de modelos de Processamento de Linguagem Natural (PLN) e *Large Language Models* (LLM) é intrinsecamente dependente da integridade dos dados de entrada, reforçando o paradigma da IA Centrada em Dados [Ng 2021]. No entanto, a prática da engenharia de dados enfrenta o desafio da heterogeneidade de fontes ruidosas, como páginas web ricas em metadados irrelevantes e arquivos PDF cujas quebras de linha artificiais pesquisadores comprometem a estrutura lógica do texto [Bast and Korzen 2017]. Tais inconsistências, se não tratadas na origem, precipitam as chamadas “cascatas de dados”, onde falhas na extração e pré-processamento degradam irreversivelmente a eficácia do modelo final [Sambasivan et al. 2021].

O problema central reside na fragmentação do ecossistema de pré-processamento. Tipicamente, adota-se uma abordagem assistemática baseada em *scripts ad hoc*, o que resulta em alto acoplamento entre a extração e a lógica de negócio, inconsistência na normalização linguística e ausência de rastreabilidade de dados. Em documentos PDF, por exemplo, a extração ruidosa de textos dispostos em colunas e a presença de hifenizações corrompem a integridade dos *tokens*, impedindo que lematizadores identifiquem corretamente o núcleo semântico das sentenças. Esta situação é particularmente crítica no português brasileiro, onde a complexidade morfológica exige um pré-processamento rigoroso para mitigar ambiguidades [Caseli and Nunes 2026]. Nesse

cenário, a `intelli3text` é proposta como uma solução arquitetural para unificar essas etapas em um *pipeline* escalável e auditável.

## 2. Objetivo

O objetivo principal deste trabalho é apresentar a `intelli3text`, uma biblioteca *Python* baseada em arquitetura modular, projetada para unificar e padronizar as etapas de pré-processamento em fluxos de Processamento de Linguagem Natural (PLN). De forma complementar, o artefato visa demonstrar como a adoção de padrões de projeto clássicos pode mitigar a propagação de ruídos em textos de fontes heterogêneas, garantindo alta rastreabilidade e auditabilidade no preparo de dados, com foco especial nas complexidades morfológicas do português brasileiro.

## 3. Fundamentação e Metodologia

A construção deste artefato fundamenta-se na premissa de que a eficácia de modelos de linguagem depende estritamente de bases linguísticas auditáveis e tecnicamente consistentes. Para tanto, a arquitetura proposta integra a biblioteca *spaCy* na normalização morfológica e o modelo *fastText* para a identificação de idioma (LID) [Joulin et al. 2017]. A decisão técnica de realizar a LID em nível de parágrafo justifica-se pela natureza multilíngue de *corpora* contemporâneos, nos quais citações e terminologias técnicas em inglês coexistem frequentemente com o corpo do texto acadêmico em português. Esse refinamento serve para a lematização no português brasileiro, uma vez que a atribuição correta de *tags* morfossintáticas exige que o texto esteja previamente saneado de ruídos de extração, sob o risco de comprometer a integridade semântica do dado processado.

Sob a ótica metodológica, esta investigação adota a *Design Science Research* (DSR), uma abordagem orientada à criação de artefatos que visam resolver problemas práticos e cientificamente relevantes. O ciclo de desenvolvimento foi pautado no rigor proposto por [Peppers et al. 2007], partindo da identificação e motivação do problema conforme detalhado na introdução deste trabalho. A concepção da `intelli3text` foi especificamente orientada para as demandas do ecossistema *Intelli3*, sendo construída e validada no contexto da pesquisa de [Speck 2026b].

## 4. Arquitetura e Implementação

A arquitetura da `intelli3text` foi projetada sob os princípios de modularidade e separação de responsabilidades, estruturando o fluxo de dados em uma sequência lógica que garante a integridade desde a ingestão até a exportação (Figura 1). A implementação baseia-se em quatro padrões de projeto clássicos [Gamma et al. 1995]. O padrão *Strategy* é empregado para isolar os algoritmos de extração, permitindo que a biblioteca suporte novos formatos de arquivos sem alterar o núcleo do sistema.

A configuração desse *pipeline* é viabilizada pelo padrão *Builder*, que oferece ao usuário uma interface declarativa e fluida. No coração do processamento, o *Chain of Responsibility* organiza a limpeza de dados como uma sequência de elos independentes, enquanto o padrão *Factory* gerencia o carregamento dinâmico de modelos de linguagem conforme o idioma detectado, otimizando significativamente o consumo de memória RAM ao carregar recursos estritamente sob demanda [Gamma et al. 1995].

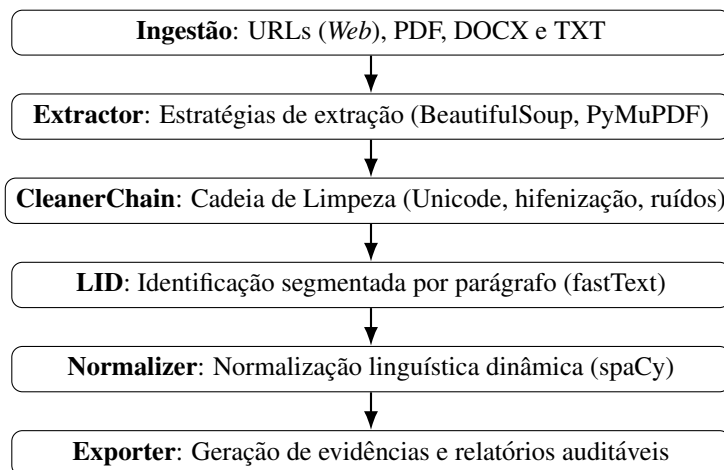


**Figura 1. Pipeline arquitetural e fluxo de transformação da biblioteca *intelli3text*.**

A organização lógica da biblioteca reflete essa separação modular, conforme ilustrado na estrutura de pacotes da Figura 2. Os módulos são dedicados a responsabilidades específicas: o *extractors* lida com as particularidades de cada formato, enquanto o *cleaners* aplica heurísticas para mitigar ruídos típicos de arquivos PDF, sendo fragmentação de colunas e hifenizações de fim de linha. A identificação de idioma e a normalização linguística residem nos módulos *lid* e *nlp*, respectivamente, assegurando que o processamento morfológico seja adaptativo e preciso.

```

src/intelli3text/
├── extractors/      (Estratégias de extração)
├── cleaners/        (Heurísticas e limpeza)
├── lid/             (Identificação via fastText)
├── nlp/             (Normalização via spaCy)
└── export/          (Relatórios e evidências)
  
```

**Figura 2. Organização modular e estrutura de pacotes do artefato.**

A *intelli3text* garante a auditabilidade do processamento, evitando “caixa-preta”, a ferramenta gera relatórios comparativos que permitem ao pesquisador visualizar o estado do dado em três estágios, sendo bruto, limpo e normalizado. Essa transparência é complementada pela normalização adaptativa, que alterna automaticamente entre modelos (Ex.: *pt\_core\_news\_lg* para o português) baseando-se na detecção de idioma em nível de parágrafo, o que previne distorções em documentos bilíngues. A aplicação prática dessa arquitetura é exemplificada na Figura 3, que demonstra a configuração de um *pipeline* capaz de reconstruir a integridade semântica de um documento PDF enquanto gera evidências para fins de auditoria e reprodutibilidade.

## 5. Validação e Resultados

A validação funcional do artefato foi conduzida por meio de um protocolo sistemático estruturado em quatro dimensões complementares. Primeiramente, a *Validação de Entrada*

---

```

from intelli3text import PipelineBuilder, Intelli3textConfig

cfg = Intelli3textConfig(
    cleaners=["ftfy", "pdf_breaks", "remove_extra_spaces"],
    nlp_model_pref="lg", # Modelos de alta precisão
    export={"pdf": {"path": "auditoria_pln.pdf"}}
)

pipeline = PipelineBuilder(cfg).build()
result = pipeline.process("https://exemplo.org/documento.pdf")

```

---

**Figura 3. Configuração da pipeline auditável.**

avaliou o processamento de fontes heterogêneas, com ênfase em páginas *Web* ruidosas e PDFs com layout em múltiplas colunas. Em seguida, a *Validação de Processamento* inspecionou a integridade Unicode e a persistência de caracteres especiais, enquanto a *Validação de Idioma* verificou a acurácia da detecção por parágrafo em documentos bilíngues. Por fim, a *Validação de Auditabilidade* conferiu a coerência e a fidelidade dos relatórios de evidências gerados pela *pipeline*. Os resultados preliminares, sintetizados na Tabela 1, demonstram que a limpeza sistemática impacta diretamente a qualidade dos dados para tarefas subsequentes de PLN.

**Tabela 1. Impacto da limpeza na densidade lexical (médias observadas)**

| Fonte           | Redução de Carac-<br>teres | Melhoria<br>Lematização | na LID Precisão |
|-----------------|----------------------------|-------------------------|-----------------|
| Web (Notícias)  | 15% a 25%                  | +12%                    | 98.2%           |
| PDF (Acadêmico) | 5% a 10%                   | +18%                    | 94.5%           |
| DOCX (Técnico)  | 2% a 5%                    | +4%                     | 97.8%           |

Qualitativamente, observou-se que o relatório de auditabilidade permitiu a detecção precoce do denominado “ruído fantasma”, fragmentos de cabeçalhos e rodapés que, embora removidos visualmente nas ferramentas de visualização padrão, permaneciam latentes nos metadados de extração convencionais. Do ponto de vista da produtividade, a adoção da biblioteca reduziu o tempo de desenvolvimento de *pipelines* iniciais em aproximadamente 60%. Este ganho é consequência direta da abstração da complexidade técnica, eliminando a necessidade de escrita e manutenção manual de expressões regulares (*regex*) complexas para limpeza de texto, e assegurando que o esforço do pesquisador seja direcionado à análise semântica e não ao saneamento básico de dados.

## 6. Conclusões e Trabalhos Futuros

A *intelli3text* atende à necessidade de infraestrutura para o pré-processamento de dados linguísticos, garantindo transparência e rastreabilidade em fluxos de PLN. A aplicação de padrões de projeto, *Strategy*, *Builder*, *Chain of Responsibility* e *Factory*, permitiu a criação de um artefato modular que desacopla a lógica de extração da normalização semântica, mitigando falhas estruturais comuns em documentos heterogêneos.

Embora eficaz no tratamento de texto nativo, a biblioteca apresenta como limitação a dependência de camadas de texto digital, não processando documentos em formato de imagem pura. Como trabalhos futuros, propõem-se: (i) a integração de motores de OCR (*Tesseract* ou *PaddleOCR*) para suporte a acervos digitalizados; (ii) a implementação de métricas de qualidade textual pós-limpeza, como *Type-Token Ratio* (TTR); e (iii) a expansão da normalização adaptativa para outros idiomas via ecossistema *Stanza*.

O artefato cumpre seu papel oferecendo à comunidade acadêmica um mecanismo auditável para a construção de *corpora* consistentes no português brasileiro ou múltiplos idiomas.

## Referências

- Bast, H. and Korzen, C. (2017). A benchmark and evaluation for text extraction from pdf. In *Proceedings of the 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 79–88. IEEE.
- Caseli, H. d. M. and Nunes, M. d. G. V., editors (2026). *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*. BPLN, 4 edition.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Honnibal, M. and Montani, I. (2017). spaCy: Industrial-strength natural language processing in python.
- Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- McKie, J. and Liu, R. (2024). Pymupdf: Python bindings for mupdf.
- Meta AI (2026). fasttext: Language identification.
- Ng, A. (2021). Mlops: From model-centric to data-centric ai. *DeepLearning.AI (Pioneering Data-Centric AI movement)*.
- Peppers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77.
- Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., and Aroyo, L. (2021). ”everyone wants to do the model work, not the data work”: Data cascades in high-stakes ai. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, New York, NY, USA. ACM.
- Speck, J. R. (2026a). intelli3text 0.2.7.
- Speck, J. R. (2026b). Ontomi: Uma ontologia fundamentada na teoria das inteligências múltiplas para classificação semântica de recursos educacionais. Master’s thesis, Programa de Pós-Graduação em Ciência da Computação, Universidade Estadual do Oeste do Paraná.