UNIVERSIDADE FEDERAL FLUMINENSE

**ALLAN PATRICK DE FREITAS SANTANA**

# Interoperabilidade Semântica de Dados

NITERÓI

2023

ALLAN PATRICK DE FREITAS SANTANA

# Interoperabilidade Semântica de Dados

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Orientador:
Bruno Lopes

NITERÓI

2023

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S231i Santana, Allan Patrick De Freitas
Interoperabilidade Semântica de Dados / Allan Patrick De Freitas Santana. - 2023.
113 f.: il.

Orientador: Bruno Lopes.
Dissertação (mestrado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2023.

1. Ontologia. 2. Interoperabilidade. 3. Microsserviços. 4. Produção intelectual. I. Lopes, Bruno, orientador. II. Universidade Federal Fluminense. Instituto de Computação.III. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Allan Patrick De Freitas Santana

Interoperabilidade semântica de dados

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Aprovada em setembro de 2023.

BANCA EXAMINADORA

Prof. Bruno Lopes - Orientador, UFF

Prof. Cláudia Nalon, UnB

Documento assinado digitalmente
gov.br DANIEL CARDOSO MORAES DE OLIVEIRA
Data: 19/10/2023 09:46:47-0300
Verifique em https://validar.iti.gov.br

Prof. Daniel de Oliveira, UFF

Niterói

2023

# Agradecimentos

Gostaria de expressar minha sincera gratidão a todas as pessoas que estiveram ao meu lado nessa jornada longa e desafiadora.

À minha namorada Tatiane, que sempre me apoiou e esteve ao meu lado, me auxiliando, me apoiando, me incentivando e oferecendo seu carinho e ajuda ao longo do caminho.

À minha mãe, que, mesmo diante de todas as dificuldades e complicações, nunca deixou de me apoiar e incentivou minhas decisões.

Ao meu pai, cuja presença e apoio foram constantes, sempre disposto a me ajudar quando necessário.

Ao meu padrasto, que sempre se dedicou a compartilhar seu conhecimento e esforçou-se para me apoiar em tudo o que precisei.

À minha tia, que esteve sempre presente e pronta para auxiliar em qualquer dificuldade que eu enfrentasse.

Ao meu orientador Bruno Lopes, cuja incrível dedicação e auxílio foram constantes ao longo de nossa colaboração.

Aos meus amigos e familiares, que sempre demonstraram carinho e ofereceram seu apoio incondicional em toda a trajetória acadêmica.

Agradeço também à Prefeitura de Niterói pelo seu apoio e parceria, que, nos bastidores, me ajudou em minha conquista.

Cada um de vocês desempenhou um papel fundamental em minha vida e sou profundamente grato por todo o suporte e incentivo que recebi ao longo do percurso. Obrigado por fazerem parte dessa conquista.

# Resumo

Em um cenário digital em constante evolução, o fluxo de informações disponíveis ao público não para de crescer. No entanto, essa expansão ocorre por meio de um cenário fragmentado, composto por diversas empresas, cada uma contribuindo de maneira única. Essa orquestração da disseminação de dados, contudo, não é desprovida de desafios. À medida que a integração de informações avança, discrepâncias e falhas podem emergir, lançando dúvidas sobre a integridade dos sistemas. O esforço para alinhar esse ecossistema diversificado pode ser complexo e demandar recursos substanciais, frequentemente apresentando desafios significativos para as empresas, por esta razão a Interoperabilidade de dados é uma solução viável, visto que é um mecanismo que visa preencher a lacuna entre formatos divergentes e fontes multifacetadas. No cerne desse paradigma está o processamento de dados heterogêneos, transformando-os em um agregado de informações unificadas. Nesse contexto, o presente estudo apresenta uma arquitetura, elaborada para coordenar o alinhamento de dados provenientes de múltiplas fontes. Este alinhamento é realizado por meio de microsserviços e possui uma conexão com a ontologia SUMO, conferindo à arquitetura a capacidade de efetuar inferências a partir dos dados unificados. Além da arquitetura uma ferramenta com uma interface foi construída para validar a proposta apresentada.

**Palavras-chave**: Ontologia, Interoperabilidade, Microsserviços.

# Abstract

In a constantly evolving digital world, the flow of information available to the public continues to grow. However, this expansion takes place through a fragmented scenario, composed of diverse companies, each contributing in a unique manner. This orchestration of data dissemination, however, is not devoid of challenges. As the integration of information progresses, discrepancies and failures may emerge, casting doubts on the integrity of systems. The effort to align this diverse ecosystem can be complex and demand substantial resources, often presenting significant challenges for companies. For this reason data interoperability is a viable solution since it is a mechanism aimed at bridging the gap between divergent formats and multifaceted sources. At the core of this paradigm lies the processing of heterogeneous data, transforming it into a unified aggregate of information. In this context, the present study introduces an architecture designed to coordinate the alignment of data from multiple sources. This alignment is achieved through microservices and has a connection to the SUMO ontology, granting the ability to make inferences from the unified data. In addition to the architecture, a tool with an interface has been developed to validate the presented proposal.

**Keywords**: Ontology, Interoperability, Microservice.

# Lista de Figuras


1 Ontologia de pessoa . . . . . 22

2 Ontologia de pessoa . . . . . 23

3 Grafo de livro na WordNet . . . . . 26

4 Estrutura da SUMO . . . . . 28

5 Etapas da arquitetura . . . . . 41

6 Fluxo da ferramenta . . . . . 47

7 Fluxograma do alinhamento de pessoas. . . . . 56

8 Fluxograma do alinhamento de Dnas. . . . . 58

9 Fluxograma do alinhamento de cidades. . . . . 61

10 Fluxograma do alinhamento de pessoas. . . . . 66

11 Fluxograma do alinhamento de dnas. . . . . 69

12 Fluxograma do alinhamento de cidades. . . . . 75

13 Página de Home da interface . . . . . 92

14 Página de Home da interface preenchida . . . . . 93

15 Página de seleção de arquivos da interface . . . . . 94

16 Página de seleção de arquivos da interface preenchida . . . . . 94

17 Página de seleção de consultas da interface . . . . . 95

18 Página de seleção de consultas da interface preenchida . . . . . 95

19 Página de seleção de inferências da interface . . . . . 96

20 Página de seleção de inferências da interface preenchida . . . . . 97

21 Página de processamento da interface . . . . . 98

22 Página de download da interface . . . . . 98

23 Página de repetição de consultas da interface . . . . . 99
24 Página de repetição de inferências da interface . . . . . 99
25 Página de processamento da interface . . . . . 100
26 Página de download da interface . . . . . 100
27 Página de escolha de processo da interface . . . . . 101
28 Página de seleção de arquivos do Deep Matcher da interface . . . . . 102
29 Página de processamento do Deep Matcher da interface . . . . . 102
30 Página de download do Deep Matcher da interface . . . . . 103
31 Página de seleção de arquivos do Pandas da interface . . . . . 104
32 Página de seleção de consultas do Pandas da interface . . . . . 104
33 Página de processamento do Pandas da interface . . . . . 105
34 Página de download do Pandas da interface . . . . . 105
35 Página de seleção de arquivos do SUMO da interface . . . . . 106
36 Página de seleção de inferências do SUMO da interface . . . . . 107
37 Página de processamento do SUMO da interface . . . . . 107
38 Página de download do SUMO da interface . . . . . 108

# Lista de Tabelas


1 Bigbase Pessoa . . . . . 79

2 Bigbase Dna . . . . . 80

3 Bigbase Cidade . . . . . 80

4 Objeto da SUMO Pessoa . . . . . 83

5 Objeto da SUMO Dna . . . . . 84

6 Objeto da SUMO Cidade . . . . . 84

7 Consulta Pessoa . . . . . 87

8 Consulta Dna . . . . . 88

9 Consulta Cidade . . . . . 88

# Sumário


**1 Introdução** **12**

**2 Ferramental** **14**

2.1 Ferramental lógico . . . . . 14

2.1.1 Lógica proposicional . . . . . 14

2.1.2 Lógica de primeira ordem . . . . . 15

2.1.3 Lógica de descrição . . . . . 16

2.1.4 Ontologia . . . . . 21

2.1.4.1 OWL . . . . . 24

2.1.4.2 WordNet . . . . . 25

2.1.4.3 SUMO . . . . . 27

2.2 Ferramental tecnológico . . . . . 28

2.2.1 Dicionário de sinônimos . . . . . 28

2.2.2 Distância de edição . . . . . 29

2.2.3 Tradução . . . . . 31

2.2.4 Deep Matcher . . . . . 32

2.3 Pandas . . . . . 36

**3 Ferramenta para alinhamento de dados** **38**

3.1 Primeiro ciclo . . . . . 38

3.2 Segundo ciclo . . . . . 38

3.3 Terceiro ciclo . . . . . 39

3.4 Desafios . . . . . 39
3.5 Arquitetura . . . . . 40
3.5.1 Conversão . . . . . 41
3.5.2 Equivalência de campos . . . . . 42
3.5.3 Equivalência de propriedades . . . . . 43
3.5.4 Geração de Resultados . . . . . 43
3.6 Algoritmo . . . . . 44
3.6.1 Conversão . . . . . 48
3.6.2 Equivalência de campos . . . . . 53
3.6.2.1 Exemplo de alinhamento dos campos . . . . . 54
3.6.2.2 Exemplo de alinhamento dos campos . . . . . 57
3.6.2.3 Exemplo de alinhamento dos campos . . . . . 59
3.6.3 Equivalência de indivíduos . . . . . 63
3.6.3.1 Exemplo de alinhamento de indivíduos . . . . . 64
3.6.3.2 Exemplo de alinhamento de indivíduos . . . . . 67
3.6.3.3 Exemplo de alinhamento de indivíduos . . . . . 71
3.6.4 Geração de Resultados . . . . . 79

**4 Interface** **90**
4.1 Deep Matcher . . . . . 101
4.2 Pandas . . . . . 103
4.3 SUMO . . . . . 106

**5 Conclusões e trabalhos futuros** **109**

**REFERÊNCIAS** **112**

# 1 Introdução

A interoperabilidade de dados (CANDELA; CASTELLI; PAGANO, 2013) é o processo de lidar com informações provenientes de múltiplas fontes ou em formatos diversos, com o objetivo de entregá-las ao usuário final de maneira unificada. Em outras palavras, é a capacidade de acessar bases de dados separadas de forma transparente, sem que o usuário precise ter conhecimento da complexidade da estrutura dos dados.

Devido ao crescente volume de dados distribuídos, alcançar a interoperabilidade torna-se uma tarefa desafiadora, uma vez que poucos sistemas armazenam informações estruturadas de forma consistente. Adicionalmente, os dados podem ser nomeados de forma semelhantes como, por exemplo, duas representações de um campo de telefone onde sua semântica pode variar. Isso pode impactar tanto a maneira como os resultados são interpretados pelos sistemas quanto o tratamento semântico e/ou sintático dos dados, podendo exigir a avaliação de um especialista para verificar explicitamente o seu significado.

O alinhamento de dados é o processo de combinar uma ou várias fontes de dados diferentes. Isso envolve a integração de informações de diversas origens para criar um conjunto de dados unificado e coeso. O objetivo principal do alinhamento de dados é criar uma visão única e abrangente que permita uma análise mais eficaz e uma compreensão mais completa das informações. Esse processo é fundamental em várias áreas, desde a consolidação de informações empresariais até a pesquisa científica, onde a fusão de dados de várias fontes é essencial para obter informações significativas e tomar decisões embasadas.

Ontologias são especificações que visam conceitualizar um domínio de conhecimento (LUCIANO; FLORIDI, 2003). Estas utilizam lógica de descrição para representar os sistemas de forma taxonômica e podem ser utilizadas para integrar os dados. No entanto, a utilização de ontologias apresenta desafios significativos, uma vez que pode haver várias abordagens para representar o conhecimento relacionado aos dados a serem estudados, resultando em diferentes concepções do mesmo domínio. Essa diversidade pode levar a

inconsistências na estratégia usada para alinhar as bases de dados com a ontologia.

A tarefa de alinhar bases de dados é complexa, exigindo a conciliação de diversas perspectivas sobre os dados para que não aconteça erros neste processo. Essas diferenças na representação e conceituação podem impactar a integração de dados e a interpretação dos resultados, tornando essencial um cuidadoso processo de alinhamento para garantir a consistência e a correta interpretação do conhecimento representado. O alinhamento adequado das ontologias é fundamental para obter resultados confiáveis e úteis em aplicações que dependem dessa estrutura conceitual.

Esta dissertação visa proporcionar uma ferramenta e arquitetura para tratar o problema de alinhamento sobre dados vindos de diferentes fontes, com este foco o presente trabalho apresenta uma arquitetura baseada em microsserviços e ontologia para alcançar a interoperabilidade de dados de forma semiautomática e expansível. Para isso, foi desenvolvida uma ferramenta que utiliza múltiplos serviços, gerando diversas saídas de dados e uma ontologia alinhada com o conjunto de bases disponíveis. Além disso, uma interface foi criada para facilitar o uso da ferramenta.

A estrutura desta dissertação segue a seguinte forma: O Capítulo 2 apresenta a parte teórica e o ferramental tecnológico utilizados. O Capítulo 3 aborda a arquiteta e ferramenta desenvolvida e seu funcionamento. No Capítulo 4, apresentamos a interface criada e como ela opera. O Capítulo 5 apresenta as conclusões alcançadas ao longo do projeto.

# 2 Ferramental

## 2.1 Ferramental lógico

Esta seção introduz o conjunto de ferramentas lógicas utilizadas nesta dissertação.

### 2.1.1 Lógica proposicional

A Lógica Proposicional é um ramo da lógica que se dedica ao estudo de proposições e suas relações. Seu modelo matemático permite o raciocínio de sentenças lógicas chamadas proposições. Nesse modelo, as proposições podem ser verdadeiras ou falsas, porém não podem ter ambos os valores simultaneamente. Além disso, é possível combinar diferentes proposições para formar sentenças mais complexas (AHO; ULLMAN, 1994). Um exemplo de aplicação da Lógica Proposicional é a modelagem da afirmação "Arthur gosta de jogos de tabuleiro", que pode ser representada por $A \equiv$ "Arthur gosta de jogos de tabuleiro", em que $A$ é um símbolo proposicional que denota a proposição.

A sintaxe da Lógica Proposicional é definida por sentenças proposicionais representadas por fórmulas, que podem ser atômicas ou moleculares. As fórmulas atômicas correspondem a proposições simples que não possuem conectivos lógicos, como por exemplo, "Rafael é alto" ou "Tatiane é bonita".

A lógica proposicional utiliza regras de inferências para realizar raciocínios sobre as proposições. Estas regras de inferência são diretrizes lógicas que permitem deduzir novas proposições com base em proposições existentes, por exemplo, dadas as proposições "Se João estudar para o exame, ele terá sucesso." e "João está estudando para o exame." logo deduz-se que "João terá sucesso no exame".

Por outro lado, as fórmulas moleculares são formadas por proposições conectadas por conectivos lógicos, tais como "e", "ou" e "não". Esses conectivos são utilizados para combinar proposições e formar fórmulas mais complexas. Por exemplo, temos a fórmula molecular "Rafael e Victor são magros" ou "Eduardo ou Mônica possuem filhos". Formal-

mente, a linguagem da Lógica Proposicional pode ser dada pela Definição 1.

**Definição 1** (Lógica Proposicional)**.** *A linguagem de lógica proposicional pode ser dada conforme a seguir*

*Símbolos proposicionais* $p, q, r, \ldots$ *;*

*BNF definida por:*

$$\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid \neg\varphi \mid p,$$

### 2.1.2 Lógica de primeira ordem

A lógica de primeira ordem, também conhecida como lógica de predicados, é uma extensão da lógica proposicional que permite a representação de quantificadores e predicados (RUSSELL; NORVIG, 2016). Enquanto que a lógica proposicional lida apenas com proposições simples e conectivos lógicos, a lógica de primeira ordem possibilita a expressão de relações complexas entre indivíduos e a formulação de proposições quantificadas, como "para todo x" e "existe um x".

A principal diferença entre a lógica proposicional e a lógica de primeira ordem está na capacidade de representação. A lógica proposicional lida apenas com proposições simples e operadores lógicos, sendo limitada a expressar relações entre proposições. Em contraste, a lógica de primeira ordem é mais expressiva, permitindo a representação de quantificadores, variáveis e predicados, o que a torna adequada para expressar relações complexas entre objetos e conceitos do mundo real. Em contraparte a lógica proposicional é usada para raciocinar sobre afirmações simples, a lógica de primeira ordem é usada para modelar situações mais complexas e expressar detalhes precisos em áreas como inteligência artificial, matemática e filosofia. A linguagem da lógica de primeira ordem pode ser formalmente dada pela Definição 2.

**Definição 2** (Lógica de primeira ordem)**.** *A linguagem da lógica de primeira ordem pode ser descrita conforme BNF a seguir.*

$$\varphi ::= \psi(\eta) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x\varphi \mid \exists x\varphi,$$

*tal que* $\eta ::= \eta, \eta \mid x \mid f(\eta)$ *onde* $\psi$ *é um predicado,* $f$ *é uma função e* $x$ *é uma variável ou constante.*

### 2.1.3 Lógica de descrição

A lógica de descrição é uma família de linguagens formais que representam domínios de forma estruturada e organizada. Esses domínios representam conhecimentos e suas características, como informações estruturadas sobre pessoas, leis, comidas e economia. A lógica de descrição compartilha semelhanças com a lógica de primeira ordem, mas também apresenta diferenças distintas.

A diferença entre a lógica de descrição e a lógica de primeira ordem reside na sua expressividade e aplicação. A lógica de descrição é um formalismo utilizado principalmente para representar conhecimento sobre classes de objetos e suas relações, mas é limitada em termos de expressividade, não permitindo a representação de quantificadores ou funções complexas. Por outro lado, a lógica de primeira ordem é mais expressiva e poderosa, permitindo a representação de quantificadores, variáveis, funções e predicados, tornando-a adequada para modelar relações complexas e expressar detalhes precisos em diversas áreas.

Essa distinção é importante porque a decidibilidade da lógica de descrição permite a aplicação de algoritmos para resolver problemas de raciocínio em domínios representados nessa lógica. Por outro lado, a lógica de primeira ordem é mais expressiva e permite uma representação mais abrangente de conhecimentos e raciocínios.

A sintaxe de uma lógica de descrição é representado por meio de conceitos de descrição, cujas expressões são formuladas a partir de conceitos atômicos contendo predicados unários, bem como conceitos de regras que envolvem predicados binários (FRANZ et al., 2007). Um vocabulário na lógica de descrição é composto por dois componentes principais: *TBox* e *ABox*.

O *TBox* representa a parte terminológica, que contém conhecimento sobre as estruturas e relações do domínio em questão. Ele descreve os conceitos, as hierarquias conceituais, as restrições de propriedades e as relações entre os conceitos. Essa parte da lógica de descrição permite a definição de classes de objetos, suas características e as restrições impostas a eles.

Por outro lado, a *ABox* é a parte assertiva da lógica de descrição, que representa o conhecimento sobre situações concretas do domínio. Ela consiste em afirmações individuais, onde os indivíduos específicos do domínio são relacionados aos conceitos e propriedades definidos no *TBox*. A *ABox* permite representar fatos concretos e instâncias do domínio em estudo.

A combinação do *TBox* e da *ABox* forma a KB, ou base do conhecimento, que contém todas as informações sobre o domínio em questão, tanto a parte terminológica quanto os fatos assertivos. Essa base de conhecimento é utilizada para realizar inferências e raciocínio na lógica de descrição.

A definição dos conceitos em uma lógica de descrição é formalizada pela Definição 3, que especifica as regras e estruturas para a representação dos conceitos dentro do formalismo da lógica de descrição.

**Definição 3** (Lógica de descrição)**.** *A linguagem da lógica de descrição básica chamada AL (linguagem atributiva) pode ser descrita conforme a seguir:*

*Dados $C$ e $D$ conceitos complexos, $A, B$ conceitos atômicos e $R$ papéis atômicos;*

*O conjunto de conceitos de descrição sobre $C$, $D$ e $R$ pode ser definido indutivamente por:*

$$
\begin{aligned}
C,\ D \ \rightarrow\ & A \mid (Conceito\ Atômico) \\
& \top \mid (Conceito\ Universal) \\
& \bot \mid (Conceito\ de\ Contradição) \\
& \neg A \mid (Negação\ atômica) \\
& C \sqcap D \mid (Interseção) \\
& \forall R.C \mid (Restrição\ de\ valor) \\
& \exists R.\top \mid (Quantificação\ existencial\ limitada)
\end{aligned}
$$

*Na AL tem-se que a negação pode ser aplicada apenas em conceitos atômicos.*

Um exemplo de domínio relacionado a professores em uma universidade pode ser representado pelo exemplo 4. No *TBox* desse exemplo, teríamos uma descrição como "um professor é uma pessoa que ministra aulas", definindo assim a relação entre a entidade "professor" e a entidade "pessoa". Já na *ABox*, teríamos uma afirmação como "Frederico é uma pessoa", estabelecendo a informação específica de que Frederico é uma instância da entidade "pessoa".

Essa representação exemplifica como o *TBox* e a *ABox* são utilizados na lógica de descrição para descrever e representar conhecimento em um domínio específico. O *TBox* define as estruturas conceituais e as relações entre os conceitos, enquanto o *ABox* fornece informações concretas sobre as instâncias e os fatos assertivos relacionados a esse domínio.

Essa abordagem permite uma representação precisa e organizada do conhecimento em diversos domínios, facilitando o raciocínio lógico e a inferência sobre o domínio em questão. A lógica de descrição oferece uma base sólida para a modelagem de conhecimento e o desenvolvimento de sistemas de raciocínio automatizados em uma variedade de aplicações.

**Definição 4** (Exemplo de domínio de professores)**.** *Dado os conceitos a seguir:*

*Professor, Pessoa, Aluno como conceitos atômicos;*

*temOrientandos e estáEmAula como papéis atômicos;*

*Pode-se realizar as seguintes construções:*

- $Professor \sqcap Pessoa$
- $Aluno \sqcap Pessoa$
- $Aluno \sqcap \ \forall estáEmAula.Professor$
- $Professor \sqcap \ \exists temOrientandos.T$

A definição formal semântica de um conceito em uma linguagem de descrição $AL$ é estabelecida por meio de uma interpretação $\mathcal{I}$. Essa interpretação consiste em um conjunto não-vazio chamado domínio da interpretação, denotado por $\Delta^{\mathcal{I}}$. Além disso, é utilizada uma função de interpretação que associa a cada conceito atômico $A$ um conjunto $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. Da mesma forma, para cada relação binária $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Com base nessas definições, a semântica da lógica de descrição pode ser formalmente apresentada na Definição 5. Essa definição estabelece as regras e estruturas que regem a interpretação dos conceitos em uma linguagem de descrição $AL$.

**Definição 5** (Definição semântica de uma $\boldsymbol{AL}$)**.** *A função é estendida para conceitos de descrição abaixo de forma indutiva:*

$T^{\mathcal{I}} = \Delta^{\mathcal{I}}$

$\bot^{\mathcal{I}} = \varnothing$

$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$

$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \ \cap D^{\mathcal{I}}$

$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b. \ (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$

$(\exists R.T)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a, b) \in R^{\mathcal{I}}\}$

*Tem-se que dois conceitos $C$ e $D$ são equivalentes quando $C \equiv D$ se $C^{\mathcal{I}} = D^{\mathcal{I}}$ para todas as interpretações de $\mathcal{I}$*

Essa abordagem semântica fornece as bases para a interpretação dos conceitos em uma linguagem de descrição, estabelecendo precisamente os significados e relações entre os conceitos. Com isso, é possível realizar inferências e raciocínios lógicos sobre os dados e informações representados nessa linguagem. Esta definição formal semântica possibilita uma interpretação precisa e coerente dos conceitos, contribuindo para a aplicação efetiva dessa lógica em diversas áreas, como representação de conhecimento, ontologias e sistemas de raciocínio automatizado.

Além da linguagem de descrição $AL$ padrão, existe uma família de linguagens chamadas $AL$ ampliadas, nas quais são adicionados construtores adicionais à linguagem $AL$ com o intuito de torná-la mais expressiva. Essas adições são especificadas na Definição 7, à qual fornece uma base formal para compreender e utilizar corretamente os construtores adicionais e suas regras semânticas na linguagem ampliada.

**Definição 6** (Adições a $\boldsymbol{AL}$)**.** *Abaixo tem-se as fórmulas adicionadas pela família de linguagens AL:*

*O conceito de união $U$ pode ser representado por $C \sqcup D$*

*A quantificação existencial completa $\mathcal{E}$ é definido como $\exists R.C$*

*A restrição por número $N$ é escrita por $\geqslant nR$ como uma restrição de pelo menos maior que e $\leqslant nR$ como uma restrição de pelo menos menor que, onde $n$ representa um valor não negativo*

*A negação de conceitos arbitrários $C$ é descrito como $\neg C$*

**Definição 7** (Adições a $\boldsymbol{AL}$)**.** *Abaixo tem-se as representações das fórmulas adicionadas pela família de linguagens AL:*

*$C \sqcup D$ é interpretado como $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} U D^{\mathcal{I}}$*

*$\exists R.C$ é interpretado como $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$, contudo está notação é diferente de $\exists R.\top$ onde pode-se aplicar conceitos arbitrários no escopo existencial*

*As restrições de maior que e menor que são interpretadas por* $(\geqslant nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | |\{b|(a,b) \in R^{\mathcal{I}}\}| \geq$ *e* $(\leqslant nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b|(a,b) \in R^{\mathcal{I}}\}| \leq n\}$ *respectivamente, onde* $|.|$ *representa a cardinalidade de um conjunto.*

$\neg C$ *é interpretado como* $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash CI$.

As linguagens $AL$ ampliadas proporcionam recursos adicionais que vão além da linguagem básica $AL$, permitindo uma representação mais abrangente e complexa de conhecimento em domínios específicos. Os construtores adicionais introduzidos na família $AL$ ampliada podem incluir, por exemplo, operadores de união, interseção, negação e restrições quantificadas.

A lógica de descrição empregada neste projeto é uma variante da lógica AL, conhecida como ALC (*Attributive Language with Complements*). ALC, além de possuir as características básicas da lógica AL, permite o uso de complementos em qualquer conceito, não se limitando apenas a complementos de conceitos atômicos.

As expressões de conceitos em ALC são bastante flexíveis e abrangentes. Podem incluir nomes de conceitos, interseção de conceitos, união de conceitos, complementos, quantificadores existenciais e universais, bem como nomes individuais.

Essa extensão da lógica de descrição oferece uma gama mais ampla de recursos e possibilidades para a representação sobre conceitos. Ela permite uma descrição mais precisa e expressiva de relações entre objetos e propriedades dos conceitos envolvidos.

As terminologias desempenham um papel fundamental na lógica de descrição ao estabelecer as relações entre conceitos e papéis. Nesse contexto, as definições devem ser tratadas como axiomas específicos e as terminologias são definidas como conjuntos de tais definições. Além disso, os conceitos atômicos podem ser utilizados como abreviações ou nomes para conceitos complexos, proporcionando uma linguagem mais concisa e compreensível.

Para facilitar o entendimento e o uso dessas terminologias, é importante estabelecer uma base sólida de definições e axiomas específicos. Essas definições servem como alicerces para a construção de conceitos complexos. Estas terminologias estão na Definição 8.

**Definição 8** (Terminologias)**.** *Tendo $C$ e $D$ conceitos e $R$ e $S$ papéis, nos casos gerais as terminologias terão as seguintes formas:*

$C \sqsubseteq D \quad (R \sqsubseteq S)$

$C \equiv D \quad (R \equiv S)$

A satisfatibilidade dessas terminologias se dá por uma interpretação $I$ a qual satisfaz uma inclusão $C \sqsubseteq D$ se $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ e ela satisfaz a igualdade $C \equiv D$ se $C^{\mathcal{I}} = D^{\mathcal{I}}$. Dado $T$ um conjunto de terminologias tem-se que $I$ satisfaz $T$ se $I$ satisfaz cada elemento de $T$. Caso $I$ satisfaça uma terminologia tem-se que $I$ é um modelo desta terminologia. Dois conjuntos de axiomas são equivalentes se eles possuem os mesmos modelos.

Quando um conjunto de axiomas é satisfeito por uma interpretação $I$, isso significa que $I$ é compatível com todas as afirmações presentes nos axiomas. $I$ é um modelo para esses axiomas, ou seja, os axiomas são verdadeiros sob a interpretação $I$. Essa equivalência entre conjuntos de axiomas é importante para a análise e a comparação de diferentes teorias expressas na lógica de descrição.

### 2.1.4 Ontologia

Na lógica de descrição, os domínios do conhecimento são frequentemente referidos como ontologias. Essas ontologias são representações formais que descrevem a nomenclatura, definição, propriedades e relações das informações e entidades em um domínio específico. No campo da computação, as ontologias são modelos conceituais utilizados para representar dados e tentar modelar um determinado domínio de conhecimento. Esses modelos são compostos por classes, que representam categorias de entidades, atributos que descrevem as propriedades dessas entidades no domínio, e relações que estabelecem conexões entre as classes. Essas propriedades incluem informações sobre o comportamento do domínio e as restrições impostas aos dados. Por exemplo, em um domínio de professores de uma faculdade, as propriedades podem incluir a pertinência dos professores à faculdade e a condição de pessoas serem professores ou não.

Ao comparar ontologias com bancos de dados, podemos considerar as ontologias como uma abstração dos modelos de banco de dados, concentrando-se na modelagem das informações sobre indivíduos, suas propriedades e relações. As ontologias são normalmente especificadas usando linguagens de ontologia, como a OWL (*Web Ontology Language*). Essa semelhança permite que as ontologias sejam utilizadas para integrar bancos de dados heterogêneos, facilitando a interoperabilidade entre sistemas distintos através de uma interface baseada na ontologia. Dessa forma, as ontologias fornecem uma camada de abstração que permite uma compreensão comum e compartilhada dos dados, independentemente das diferenças nos modelos de banco de dados subjacentes.

As ontologias fornecem um vocabulário que permite aos usuários fazer inferências sobre os domínios representados por elas. Essas inferências podem ser expressas por meio de perguntas feitas à ontologia, como "quais professores trabalham em uma determinada faculdade" ou "quais professores utilizam o sistema de transporte disponibilizado pela faculdade", com base na ontologia que descreve os professores de uma faculdade. Essa característica permite que os usuários realizem inferências sobre o sistema sem precisar conhecer a estrutura dos dados subjacentes. Para ilustrar esse conceito, um exemplo simples de uma ontologia e algumas das inferências que podem ser obtidas a partir dela estão ilustradas na Figura 1.

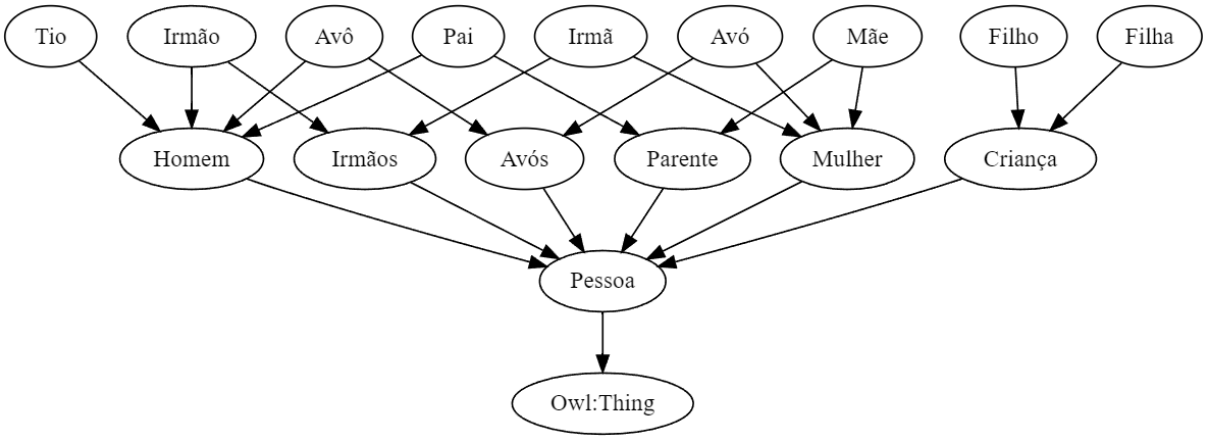


Figura 1: Ontologia de pessoa

Neste exemplo, é ilustrado uma ontologia que representa o conceito de pessoa. Nessa ontologia, os vértices representam as características de uma pessoa de acordo com essa concepção. Ela define que uma pessoa pode ser do sexo masculino, ser irmã de alguém, ser avó de outra pessoa, entre outras características. É importante destacar que essas características não são mutuamente exclusivas. Portanto, uma mulher pode, simultaneamente, ter irmãos e ser avó

Uma ontologia é uma representação formal de conhecimento em um domínio específico, estruturada como um grafo composto por nós que representam conceitos ou entidades e arestas que denotam relações entre eles. Essa representação gráfica facilita a compreensão das hierarquias, relações e restrições dentro do domínio, permitindo consultas, inferências e aplicação em diversas áreas, como web semântica, sistemas de recomendação e processamento de linguagem natural.

Este exemplo ilustra uma ontologia que representa o domínio de pessoas suas características e relações, em uma estrutura de grafo, na qual os vértices são representados por

conceitos e as arestas por relacionamentos hierárquicos e propriedades entre conceitos. A partir dessa ontologia, é possível realizar várias inferências, como as descritas a seguir.

- Descobrir as pessoas que são parentes de determinado indivíduo.
- Localizar as pessoas que possuem filhos.
- Determinar quais pessoas são mulheres.
- Descobrir a arvore genealógica de um determinado indivíduo.
- Localizar todos os filhos de duas pessoas.

Essas inferências são possíveis graças à estrutura e às relações definidas na ontologia, que permitem extrair informações relevantes sobre as pessoas e seus atributos. Para uma visualização mais clara das relações na ontologia de pessoa, é possível consultar o Diagrama 2. Nesse diagrama, é possível observar as diferentes entidades, como "Parente", "Pessoa" e "Mãe", e as relações entre elas, como "Possui filhos". Essas relações estabelecem conexões significativas entre os elementos da ontologia, permitindo a realização de inferências e o entendimento do contexto de forma mais intuitiva.

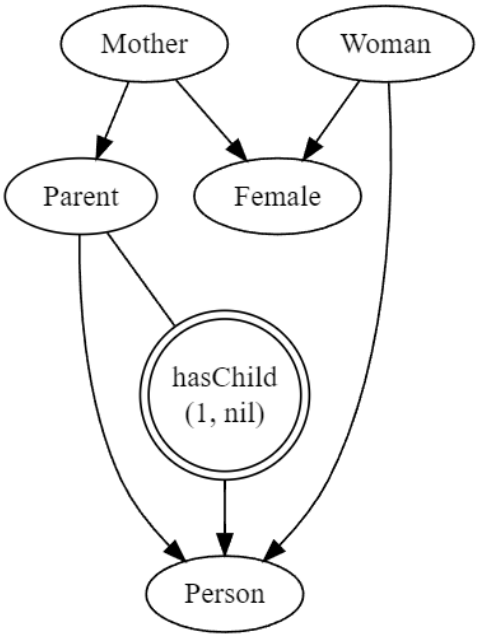


Figura 2: Ontologia de pessoa

]

Neste grafo, os vértices representam conceitos, enquanto as setas maiores indicam subconceitos. Podemos observar que "Mãe" é um subconceito de "Parente" e de "Feminina", enquanto "Mulher" é um subconceito de "Feminina" e de "Pessoa". A relação "PossuiFilhos" é representada por uma seta atravessando a relação, indicando que todo "Parente" possui pelo menos um filho. A nomenclatura (1, *NIL*) indica que todo "Parente" possui pelo menos um filho, e não há limite para o número de filhos que uma pessoa pode ter. Essa representação gráfica auxilia na compreensão das relações entre os conceitos e suas características na ontologia. As fórmulas que descrevem essa ontologia de pessoa estão descritas a seguir.

**Parente:** Parente $\equiv$ Pessoa

**Mãe:** Mãe $\equiv$ Parente e Feminina

**Mulher:** Mulher $\equiv$ Feminina e Pessoa

**Possui Filhos:** PossuiFilhos $\equiv \exists$temFilho.Pessoa (1, *NIL*)

Essas fórmulas representam as relações hierárquicas e as propriedades definidas no seu grafo de ontologia, esclarecendo as conexões entre os conceitos e suas características.

#### 2.1.4.1 OWL

OWL (*Web Ontology Language*) é uma linguagem de marcação desenvolvida pelo *World Wide Web Consortium* (W3C) (SZEREDI et al., 2014) para representar ontologias. A sintaxe principal da OWL é baseada no formato RDF (*Resource Description Framework*), que é uma extensão do XML (*Extensible Markup Language*). No entanto, as ferramentas que utilizam a OWL também podem suportar outras sintaxes, como OWL/XML, *Functional Syntax*, *Manchester Syntax* e *Turtle*. As ontologias criadas em OWL podem ser compartilhadas e reutilizadas por aplicações web, permitindo a interoperabilidade entre diferentes sistemas e domínios de conhecimento.

A OWL foi projetada para ser aplicável em diversas áreas, incorporando recursos para lidar com incertezas e ambiguidades na representação do conhecimento. Essa capacidade permite a representação de conceitos como "pode ser", "provavelmente" e "talvez", o que facilita a entrada de ontologias em formato OWL em áreas que lidam com incertezas, como a medicina. Além disso, a linguagem OWL pode ser empregada em linguagens descritivas para aprofundar a compreensão do significado das informações, oferecendo recomendações

precisas e possibilitando inferências sobre quais conceitos estão relacionados de maneira semelhante às informações desejadas. Esses recursos tornam a OWL uma ferramenta poderosa para representar e raciocinar sobre o conhecimento em diversos domínios.

A OWL é frequentemente combinada com outras ontologias na web, como a WCO (*Web Conceptual Ontology*) e a WRO (*Web Resource Ontology*), para ampliar sua expressividade na geração de modelos. Essas combinações visam integrar as ontologias criadas pela OWL aos conceitos e recursos já existentes na web, permitindo uma maior diversidade na integração de conhecimento. Ontologias desenvolvidas em OWL são amplamente utilizadas em sistemas de inteligência artificial, mineração de dados, recuperação de informações, gerenciamento de conhecimento e sistemas semânticos. Uma das maiores ontologias baseadas em OWL é a SUMO (*Suggested Upper Merged Ontology*), que é atualmente a maior ontologia formal de acesso gratuito disponível(SUMO. . . , s.d.). Essas aplicações e recursos tornam a OWL uma ferramenta valiosa para a representação e o uso eficiente do conhecimento em diversas áreas.

#### 2.1.4.2 WordNet

A WordNet é um banco de dados léxico gratuito em inglês que representa substantivos, verbos, adjetivos e advérbios organizados em grupos de sinônimos, refletindo conceitos léxicos (MILLER, 1995). Essa ferramenta baseia-se na teoria linguística cognitiva, que postula que as relações entre objetos e conceitos do mundo real assemelham-se às relações mentais estabelecidas pelas pessoas (FELLBAUM, 1998). A WordNet utiliza um grafo relacional para representar conceitos e suas relações, em que cada nó representa um conceito definido por um sinônimo cognitivo denominado "synset", e as arestas conectam os nós para representar as relações entre eles. Essas relações são classificadas em categorias como sinônimos, antônimos, hiperônimos, hipônimos, merônimos, entre outras. A WordNet desempenha um papel essencial em várias áreas, como processamento de linguagem natural, recuperação de informações e pesquisa em ciência cognitiva, fornecendo um recurso valioso para a compreensão e o estudo dos termos e seus relacionamentos.

Os *synsets* na WordNet representam o significado de uma palavra, sendo que cada *synset* contém uma ou mais palavras sinônimas e uma definição associada. Esses *synsets* também possuem relações estabelecidas entre si, como ligações *hypernym* (relacionados a um conceito mais geral) e *hyponym* (relacionados a um conceito mais específico), permitindo a criação de hierarquias semânticas entre os *synsets*. Um exemplo simples é a palavra "*book*" na WordNet, que pode ser visualizado na Figura 3. Nesse exemplo, são

apresentadas as palavras sinônimas e as relações hierárquicas existentes entre os *synsets* relacionados à palavra "*book*". Essa estrutura permite a compreensão e a exploração do significado das palavras e suas conexões dentro do contexto lexical.

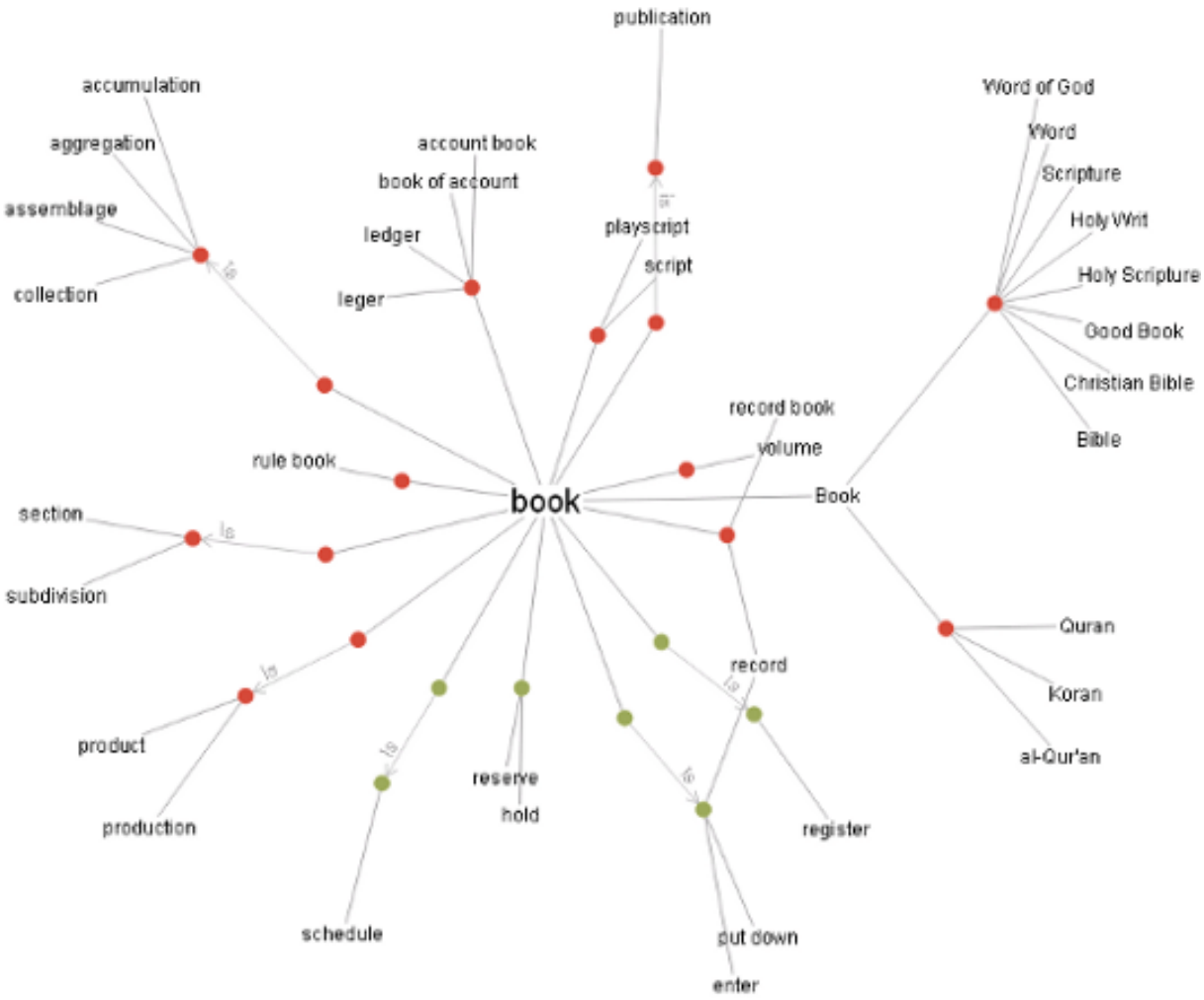


Figura 3: Grafo de livro na WordNet

A WordNet é amplamente utilizada em aplicações de processamento de linguagem natural, como recuperação de informações, tradução automática, análise de sentimentos e compreensão de texto. Por exemplo, utilizando o grafo da Figura 3, um sistema de recuperação de informações pode inferir relações semânticas da palavra "*book*". Pode-se identificar a relação semântica entre "*book*" e "*reserve*" como uma relação entre um verbo e um substantivo, assim como a relação entre "*book*" e "*enter*", classificada como uma relação entre um verbo e um substantivo, respectivamente. Além disso, esse sistema de recuperação de informações também pode inferir palavras relacionadas a "*book*", como "*rule book*", "*record book*" ou "*script*". Essas inferências permitem uma análise mais abrangente e precisa do significado da palavra "*book*" e suas conexões com outros termos

na WordNet.

A WordNet se destaca em relação a outros dicionários e bancos de dados léxicos devido ao seu sistema de rede semântica, que oferece uma maior expressividade. Ao contrário de uma simples lista de definições, a WordNet é capaz de capturar nuances de significado e estabelecer relações semânticas mais sofisticadas. Além disso, a WordNet está em constante evolução e expansão. Atualmente, está sendo exportada para outros idiomas, como alemão, francês e italiano, e recebe constantes atualizações com novos termos e significados para se manter atualizada diante das mudanças linguísticas em curso (UNIVERSITY, 2023).

#### 2.1.4.3 SUMO

A SUMO (*Suggested Upper Merged Ontology*) é atualmente a maior ontologia formal gratuita disponível (SUMO..., s.d.). Ela é uma ontologia de ordem-superior que abrange uma ampla gama de tópicos, incluindo filosofia, matemática, ciência, tecnologia e assuntos do cotidiano (NILES; PEASE, 2001). A SUMO desempenha um papel crucial em sistemas que envolvem pesquisa, solução de problemas linguísticos e raciocínio de dados. Ela é desenvolvida usando a linguagem SUO-KIF (*Knowledge Interchange Format*), uma linguagem computacional projetada especificamente para permitir o compartilhamento e a reutilização de informações entre diferentes sistemas baseados em conhecimento. Essa abordagem facilita a troca de informações e a interoperabilidade entre sistemas que utilizam a SUMO como base de conhecimento.

A SUMO (*Suggested Upper Merged Ontology*) representa sua estrutura através de conceitos simples e complexos, em que os conceitos complexos são compostos por construções que combinam os conceitos mais simples, criando assim uma hierarquia de conceitos. A relação entre os conceitos na SUMO é estabelecida por meio de relações como "é um" e "tem um", que definem como os conceitos estão relacionados entre si. Essas relações podem expressar, por exemplo, que "Livro tem páginas" ou que "Livro de regras é um tipo de livro". Um grafo que ilustra a estrutura da SUMO está apresentado na Figura 4. Esse grafo evidencia as conexões entre os conceitos e fornece uma representação visual da organização da ontologia SUMO.

[1]

A SUMO (*Suggested Upper Merged Ontology*) é uma ontologia em constante atuali-

[1] A Figura 5 foi desenvolvida com base na representação encontrada no site da SUMO, disponível no seguinte link: https://www.ontologyportal.org.

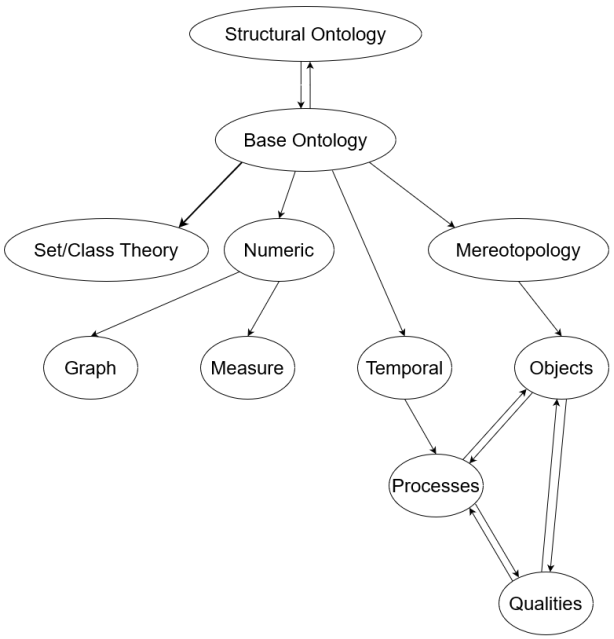


Figura 4: Estrutura da SUMO

zação, desenvolvida com o objetivo de ser amplamente utilizada por diversos sistemas e permitir a reutilização de seus dados. É a única ontologia que atualmente abrange todo o poder expressivo da WordNet e se baseia em uma vasta gama de fontes, como dicionários, enciclopédias, livros de filosofia e ciência. A SUMO possui diversas aplicações na área de inteligência artificial, abrangendo desde o processamento de linguagem natural até a inteligência artificial distribuída, robótica, planejamento, resolução de problemas e muito mais. Sua ampla utilidade demonstra a sua relevância e versatilidade como uma poderosa ferramenta para representar e raciocinar sobre o conhecimento.

## 2.2 Ferramental tecnológico

Esta seção introduz as ferramentas que serão discutidas como serviços no Capítulo 3.

### 2.2.1 Dicionário de sinônimos

Sinônimos são palavras que possuem diferentes formas ou expressões, mas compartilham o mesmo significado, como os termos “alegria” e “felicidade”. No campo da computação, a verificação de sinônimos é realizada por meio de mecanismos de processamento de linguagem natural, que permitem que os computadores processem a linguagem humana. O NLTK (*Natural Language Toolkit*) (BIRD et al., 2009) é uma biblioteca amplamente utilizada para o processamento de linguagem natural na linguagem de programação Python.

Com o auxílio do NLTK, é possível realizar tarefas como análise de texto, tokenização, identificação de sinônimos e muitas outras relacionadas ao processamento da linguagem natural.

O NLTK (*Natural Language Toolkit*) oferece uma ampla gama de recursos para o processamento de linguagem natural, abrangendo tarefas como tokenização, marcação de partes do discurso, análise de sentimentos, modelagem de linguagem baseada em probabilidades e muito mais. Através da tokenização, o NLTK consegue subdividir textos em unidades menores, conhecidas como tokens, facilitando a execução de tarefas como análise de sentimentos, classificação de texto e extração de informações.

Além disso, o NLTK disponibiliza um rico conjunto de dados já processados, que inclui corpora, palavras irrelevantes (*stopwords*) e modelos pré-treinados, o que permite utilizar a ferramenta sem a necessidade de coletar dados desde o início. Esses recursos prontamente acessíveis podem agilizar o desenvolvimento de aplicações de processamento de linguagem natural e aprimorar sua eficiência.

Na prática, o tempo de busca por sinônimos no WordNet usando o NLTK geralmente é bastante eficiente em cenários de uso típicos, principalmente para palavras individuais. Para a maioria das palavras comuns, a complexidade tende a se aproximar de um crescimento linear em relação ao número de conjuntos de sinônimos (*synsets*) associados a uma palavra.

O NLTK oferece várias funcionalidades para verificação de sinônimos, como o uso de dicionários de sinônimos pré-processados e modelos de linguística computacional. Um dos módulos disponíveis no NLTK é o módulo WordNet, onde os tokens gerados podem ser verificados em relação aos seus sinônimos presentes na lista léxica do WordNet. Para a verificação com modelos de linguística computacional, o NLTK analisa o contexto em que os tokens são utilizados, buscando palavras similares. Por exemplo, um modelo pode analisar as palavras próximas a uma determinada palavra para encontrar palavras que frequentemente ocorrem no mesmo contexto. Essas técnicas de verificação de sinônimos permitem uma abordagem mais precisa na identificação de palavras relacionadas e facilitam a compreensão do significado em diferentes contextos linguísticos.

### 2.2.2 Distância de edição

A distância de edição é uma medida que quantifica o número de operações necessárias para transformar uma sequência de caracteres em outra (BARRIÈRE, 2016). Essas operações

podem incluir inserção, exclusão ou substituição de caracteres. Por exemplo, consideremos as sequências “carro” e “carruagem”. A distância de edição entre elas é igual a 5, pois são necessárias cinco operações para transformar “carro” em “carruagem”: substituir o “o” por “u” e incluir os quatro caracteres “agem”. Outro exemplo é a distância entre “Pano” e “cano”, que é igual a 1, pois apenas uma operação é necessária: substituir a letra “p” por “c”. A distância de edição é uma medida útil em várias aplicações, como correção ortográfica, alinhamento de sequências e busca aproximada de palavras.

Existem diversas técnicas para calcular a distância de edição, entre elas o algoritmo de Hamming e o algoritmo de Levenshtein. Esses algoritmos são conhecidos por sua simplicidade e eficiência, permitindo seu uso em tempo real em muitos sistemas. No contexto desta dissertação, utilizou-se a biblioteca *textdistance* do Python (TEXTDISTANCE. . . , s.d.), que oferece uma ampla gama de algoritmos robustos para calcular a distância de edição, incluindo os mencionados anteriormente. Essa ferramenta proporciona flexibilidade e facilidade de implementação, possibilitando a escolha do algoritmo mais adequado para cada aplicação.

O algoritmo de Levenshtein é empregado para construir uma matriz que representa a distância de edição entre cada par de prefixos das sequências de origem e destino. A matriz resultante contém as distâncias de edição entre os prefixos das duas sequências, proporcionando informações valiosas sobre suas semelhanças e diferenças.

Em termos de complexidade computacional, um algoritmo que utiliza a distância de Levenshtein é geralmente classificado com complexidade O(m * n), onde “m” e “n” são os comprimentos das sequências de origem e destino, respectivamente. Isso significa que o tempo de execução do algoritmo é proporcional ao produto dos tamanhos das duas sequências, o que o torna eficiente para tarefas de comparação de strings.

A distância de Hamming é uma métrica que avalia a discrepância entre duas sequências contando o número de posições em que seus símbolos são diferentes. Essa métrica é especialmente útil ao comparar sequências que representam códigos binários, cadeias de DNA ou outras estruturas descritas por uma sequência de símbolos.

Para calcular a distância de Hamming, é comparado os símbolos correspondentes em cada posição das duas sequências e conta quantas posições apresentam símbolos diferentes. Por exemplo, a distância de Hamming entre as sequências “0000” e “1011” é 3, já que as posições 1, 3 e 4 têm símbolos diferentes. Essa métrica fornece informações valiosas sobre a dissimilaridade entre as sequências em termos de suas características simbólicas.

Em termos de complexidade computacional, um algoritmo que utiliza a distância de Hamming é geralmente classificado como tendo complexidade O(n), onde "n" é o comprimento das sequências envolvidas. Isso significa que o tempo de execução do algoritmo cresce linearmente em relação ao tamanho das sequências, tornando-o eficiente para tarefas que envolvem comparações de símbolos.

A distância de edição desempenha um papel fundamental em diversos sistemas, como processamento de linguagem natural, algoritmos de busca e classificação de texto. Sua utilidade se estende a várias aplicações, como correção de erros de digitação em palavras, sugestão de palavras alternativas com baixa distância de edição, classificação de documentos semelhantes em uma base de dados e verificação de autenticidade de documentos ou arquivos de texto por meio da comparação entre a sequência original e uma cópia ou versão modificada. Essa medida é um valioso recurso para aprimorar a precisão e eficiência de diversos sistemas que lidam com texto e informações linguísticas.

### 2.2.3 Tradução

Atualmente, a tradução de textos é uma prática amplamente difundida, impulsionada pelo crescimento da internet. Com o avanço da tecnologia, surgiram novas técnicas e ferramentas que facilitam a tradução para uma variedade de idiomas. Entre as opções disponíveis, destaca-se o *Google Translate*, a popular ferramenta de tradução de textos desenvolvida pela Google (WU et al., 2016). Essa plataforma oferece um serviço de tradução automática de linguagem natural, permitindo a tradução entre uma vasta gama de idiomas, como inglês, espanhol, francês, alemão, italiano, português, e muitos outros. O *Google Translate* se tornou uma referência no mercado de tradução, proporcionando aos usuários uma maneira prática e eficiente de obter traduções rápidas e acessíveis.

O *Google Translate* utiliza técnicas avançadas de aprendizado de máquina para realizar suas traduções. Essa abordagem consiste em treinar modelos com enormes volumes de dados previamente traduzidos, permitindo que a máquina aprenda as relações entre os idiomas e produza traduções mais precisas. À medida que o modelo adquire conhecimento, suas traduções se tornam mais acuradas. No entanto, é importante ressaltar que traduções incorretas ainda podem ocorrer em textos extensos ou complexos que exigem conhecimento especializado.

O *Google Translate* teve a sua precisão aprimorada diversas vezes durantes os anos, impulsionado pelo uso de tecnologias de aprendizado de máquina, como o modelo de tradução neural. Ao integrar técnicas avançadas de processamento de linguagem natural,

o *Google Translate* visa proporcionar uma experiência de tradução confiável e de alta qualidade para os usuários. É importante observar que a complexidade exata do algoritmo utilizado pelo *Google Translate* não é divulgada publicamente pela Google.

O *Google Translate* oferece a funcionalidade de tradução baseada em áudio, permitindo que os usuários falem em um idioma e ouçam a tradução em outro idioma. Além disso, o sistema é capaz de traduzir textos contidos em imagens. No entanto, é importante ressaltar que a precisão das traduções pode ser mais desafiadora nesses formatos.

O *Google Translate* também possui integração com outros produtos do Google, ampliando ainda mais sua funcionalidade. Por exemplo, é possível acessar o *Google Translate* diretamente no *Google Chrome*, utilizar o recurso de tradução por meio do *Google Lens*, que permite traduzir textos capturados pela câmera do dispositivo, e até mesmo utilizar a tradução por comando de voz com o auxílio do *Google Assistant*. Essas integrações proporcionam uma experiência mais completa e conveniente aos usuários que desejam realizar traduções em diferentes contextos.

### 2.2.4 Deep Matcher

O Deep Matcher é um modelo de aprendizado de máquina que emprega técnicas avançadas de processamento de linguagem natural (PLN) e aprendizado profundo para realizar alinhamentos precisos e eficientes entre conjuntos de dados ou textos (SIGMOD. . . , 2018). Esse modelo se destaca em cenários em que é necessário integrar informações de fontes diversas, mesmo quando os conjuntos de dados possuem formatos ou estruturas distintas.

Por exemplo, suponha que existam dois conjuntos de dados de professores de diferentes faculdades. Um conjunto contém informações detalhadas, como descrição das turmas, nome dos professores e número de alunos. Já o outro conjunto contém apenas o nome dos professores e o número de alunos. Utilizando o Deep Matcher, é possível realizar o alinhamento entre os dois conjuntos de dados, combinando os professores com base em seus nomes. Essa abordagem permite uma integração eficiente e automatizada das informações, facilitando a análise conjunta dos conjuntos de dados.

Quanto à complexidade do algoritmo do Deep Matcher, é importante ressaltar que essa métrica pode variar consideravelmente de acordo com uma ampla gama de fatores. Portanto, chegar a uma quantificação exata dessa complexidade pode ser desafiador devido à natureza multifacetada do modelo.

O Deep Matcher demonstra seu valor ao possibilitar a descoberta de padrões e relações

entre os dados, mesmo em cenários complexos e com múltiplas fontes de informação. Sua aplicação é ampla e pode beneficiar diversas áreas, como integração de bases de dados, enriquecimento de informações e descoberta de conhecimento em grandes volumes de dados.

O Deep Matcher é um modelo baseado em aprendizado profundo (*Deep learning*) que utiliza uma abordagem de rede neural para identificar correspondências entre os elementos de conjuntos de dados. Esse modelo é treinado em um conjunto de dados de treinamento que contém pares de elementos correspondentes e não correspondentes. Durante o treinamento, a rede neural é ajustada para maximizar a precisão na identificação dos pares correspondentes e minimizar a identificação de pares não correspondentes.

Essa abordagem de correspondência de dados tem amplas aplicações em diferentes áreas. Por exemplo, pode ser usada na integração de dados provenientes de várias fontes, permitindo combinar informações de forma mais precisa e abrangente. Além disso, o Deep Matcher é eficaz na reconciliação de registros duplicados em bancos de dados, identificação de produtos similares em catálogos de produtos e análise de textos.

Através do aprendizado profundo, o Deep Matcher é capaz de descobrir padrões complexos e relações sutis entre os elementos dos conjuntos de dados, fornecendo uma solução poderosa para tarefas de correspondência e alinhamento. Essa abordagem impulsiona a eficiência e a precisão das análises de dados, permitindo a extração de informações valiosas e significativas a partir de grandes volumes de dados.

O Deep Matcher emprega técnicas avançadas de processamento de linguagem natural para calcular a similaridade entre as entradas. Uma das abordagens utilizadas é o uso de representações vetoriais de texto, que capturam as características semânticas e contextuais das palavras. Essas representações são obtidas por meio de algoritmos que mapeiam as palavras em vetores de alta dimensionalidade com base em sua co-ocorrência em grandes conjuntos de dados de texto.

Além disso, o Deep Matcher faz uso de redes neurais convolucionais para extrair recursos relevantes das entradas. Essas redes são projetadas para identificar padrões importantes, como a presença ou ausência de certos termos e a estrutura global das sentenças. Ao analisar as características extraídas pelas redes neurais convolucionais, o modelo é capaz de avaliar a similaridade entre as entradas de maneira mais precisa e abrangente.

Essas técnicas combinadas permitem que o Deep Matcher lide com a complexidade do

processamento de linguagem natural, capturando informações sutis e contextuais presentes nas entradas. Isso resulta em uma medida de similaridade mais sofisticada e refinada, que pode ser aplicada em várias tarefas, como a comparação de documentos, a recuperação de informações e a classificação de textos.

O Deep Matcher incorpora uma arquitetura de rede neural chamada *siamese networks*, que consiste em dois ramos idênticos compartilhando os mesmos pesos e parâmetros. Essa estrutura permite que o modelo compare a similaridade entre pares de elementos dos conjuntos de dados, mesmo quando os dados possuem dimensões diferentes ou características distintas.

Além disso, o Deep Matcher utiliza a técnica de aprendizagem por pares (*pairwise learning*), que é uma abordagem de aprendizado supervisionado. Nesse processo, o modelo é treinado para aprender a realizar o alinhamento entre os pares de elementos, onde cada par é composto por uma entrada de um conjunto de dados e sua correspondente no outro conjunto. Dessa forma, o modelo aprende a avaliar a similaridade entre esses pares e a distinguir entre pares correspondentes e não correspondentes.

Essas abordagens adotadas pelo Deep Matcher permitem que o modelo seja eficaz na comparação de elementos de conjuntos de dados distintos, adaptando-se às diferenças de dimensões e características presentes nos dados. Isso torna o Deep Matcher adequado para uma variedade de aplicações, como correspondência de registros em bancos de dados, detecção de duplicatas e vinculação de informações de várias fontes de dados.

Uma vez treinado, o Deep Matcher se torna uma poderosa ferramenta para realizar alinhamentos em novos conjuntos de dados. Sua versatilidade permite que seja aplicado em diversas situações, tais como o alinhamento de registros em bancos de dados, a identificação de paridades entre itens em catálogos de produtos, a conciliação de clientes em listas de marketing e até mesmo o alinhamento entre conjuntos de dados em diferentes idiomas, superando assim as barreiras linguísticas.

Essa flexibilidade do Deep Matcher é possível devido ao seu modelo de aprendizado de máquina robusto, que aprendeu a realizar alinhamentos precisos e eficientes durante o treinamento. Ao aplicar o modelo em novos conjuntos de dados, ele é capaz de identificar correspondências e similaridades entre os elementos, trazendo melhorias valiosas para diversas áreas e setores.

Com sua capacidade de lidar com dados de diferentes estruturas e características, o Deep Matcher é uma ferramenta promissora para otimizar processos de negócios, melhorar

a qualidade dos dados e facilitar a integração de informações provenientes de múltiplas fontes. Sua aplicabilidade abrange uma ampla gama de domínios, permitindo uma maior eficiência e precisão para empresas e pesquisadores.

Uma das principais vantagens do Deep Matcher é a sua capacidade de aprender a realizar emparelhamentos de forma autônoma, dispensando a necessidade de pré-processamento manual dos dados ou de definição de regras de paridade por especialistas. Essa característica confere ao modelo uma eficiência e escalabilidade significativas, possibilitando a sua aplicação em conjuntos de dados extensos em um tempo viável.

Ao eliminar a dependência de pré-processamento manual e regras específicas, o Deep Matcher automatiza o processo de paridade, reduzindo a carga de trabalho e tornando-o mais ágil. O modelo é treinado em dados de exemplo, aprendendo a identificar padrões e similaridades entre os elementos dos conjuntos de dados. Com isso, ele é capaz de realizar emparelhamentos com base em informações aprendidas durante o treinamento, adaptando-se a diferentes tipos de dados e contextos.

Essa abordagem autônoma oferecida pelo Deep Matcher não apenas simplifica o processo de paridade, mas também traz benefícios em termos de escalabilidade. O modelo pode ser aplicado em grandes volumes de dados de forma eficiente, permitindo o processamento ágil de informações em tempo razoável. Dessa forma, o Deep Matcher se destaca como uma solução robusta e versátil para a realização de emparelhamentos em diversos domínios e cenários, impulsionando a eficiência e a produtividade em tarefas de integração de dados.

No entanto, assim como em qualquer modelo de aprendizado de máquina, o desempenho do Deep Matcher é influenciado pela qualidade dos dados de entrada e pelo processo de treinamento. Para garantir resultados satisfatórios, é essencial que os dados de treinamento sejam representativos do conjunto de dados de teste, capturando a diversidade e a variabilidade presentes. Além disso, é necessário equilibrar a quantidade de exemplos positivos e negativos durante o treinamento, a fim de evitar qualquer viés no modelo.

Outro aspecto relevante é a adequação dos hiperparâmetros do modelo. Os hiperparâmetros são configurações ajustáveis que afetam o desempenho e o comportamento do modelo. A seleção adequada desses hiperparâmetros pode otimizar a capacidade de generalização do modelo, permitindo que ele se adapte efetivamente a diferentes tarefas de alinhamento. É necessário realizar experimentos e ajustes iterativos para encontrar a combinação ideal de hiperparâmetros para cada caso de uso específico.

A garantia da qualidade dos dados de entrada, o equilíbrio dos exemplos durante o treinamento e a otimização dos hiperparâmetros são etapas cruciais para maximizar o desempenho do Deep Matcher em tarefas de alinhamento. Ao dedicar atenção a esses aspectos, é possível obter resultados mais confiáveis e precisos, impulsionando a eficácia do modelo e melhorando a qualidade dos alinhamentos realizados.

## 2.3 Pandas

O Pandas é uma biblioteca Python amplamente utilizada para análise e manipulação de dados(REBACK et al., 2022). É uma ferramenta poderosa que oferece estruturas de dados flexíveis e eficientes, projetadas para lidar com informações em formato tabular, como planilhas e bancos de dados(MCKINNEY, 2010).

Uma das estruturas de dados mais fundamentais do Pandas é o *DataFrame*, que se assemelha a uma tabela ou planilha, com colunas nomeadas e linhas indexadas. Isso torna o Pandas uma escolha popular entre cientistas de dados, analistas e engenheiros, pois permite que eles trabalhem com dados de forma intuitiva e eficaz.

O Pandas oferece uma ampla variedade de funcionalidades para importar, exportar e manipular dados. Pode-se realizar operações de filtro, agregação, junção e transformação de dados com facilidade. Além disso, o Pandas lida com valores ausentes e permite realizar cálculos estatísticos de maneira simples e direta.

O Pandas oferece uma variedade de métodos poderosos para realizar consultas e seleções de dados em suas estruturas, principalmente *DataFrames*. Essas operações permitem obter informações específicas dos dados de maneira eficaz, tornando o Pandas uma ferramenta valiosa na análise de dados.

O Pandas pode selecionar colunas específicas de um *DataFrame* simplesmente especificando os nomes das colunas entre colchetes duplos, o que permite isolar informações específicas de interesse. Além disso, é possível aplicar filtros para selecionar linhas que atendam a critérios específicos. Isso é útil para segmentar os dados com base em condições, como selecionar todas as pessoas com idade superior a 30 anos.

A combinação de filtros é uma prática comum e é facilmente realizada com operadores lógicos, como "e" (representado por &) e "ou" (representado por |). Isso permite criar consultas mais complexas que envolvem múltiplos critérios. O Pandas também fornece o método "*query*", que permite realizar consultas "*SQL-like*" nos *DataFrames*. Essa

funcionalidade facilita a execução de consultas mais elaboradas de forma legível e eficiente.

Além disso, pode-se selecionar tanto linhas quanto colunas, o que é útil para extrair informações específicas de interesse em conjuntos de dados maiores. A ordenação dos dados é outra operação comum, que pode ser realizada com o método "*sort_values*". Isso permite organizar os dados em uma ordem específica com base nas colunas desejadas.

Outros recursos do Pandas incluem a capacidade de consultar valores únicos em uma coluna, aplicar funções agregadas, como média e soma, e realizar diversas operações de pré-processamento de dados de maneira eficaz. Outro recurso notável do Pandas é a capacidade de visualização de dados, facilitando a geração de gráficos e visualizações a partir dos dados em um *DataFrame*. Isso é útil para a exploração inicial de dados e a comunicação de resultados.

# 3 Ferramenta para alinhamento de dados

A origem deste trabalho reside na união de um conjunto de serviços destinados a realizar alinhamentos em bases de dados. Esses serviços foram cuidadosamente selecionados com base em cenários comuns de alinhamento, incluindo casos de diferenças gramaticais sutis, alinhamento de palavras com significados semelhantes, integração de dados em múltiplos idiomas e a avaliação de alinhamentos gerados por algoritmos de inteligência artificial.

Com os serviços escolhidos, colaboramos em parceria com a Prefeitura de Niterói. A prefeitura compartilhou conjuntos de tabelas de dados e solicitou a execução de diversas tarefas sobre esses dados. O processo de desenvolvimento foi conduzido em ciclos, alinhados às necessidades específicas apresentadas pela prefeitura.

Essa abordagem permitiu a adaptação da ferramenta à medida que novas demandas surgiam, garantindo que a ferramenta atendesse de forma flexível e precisa às solicitações da Prefeitura de Niterói.

## 3.1 Primeiro ciclo

No primeiro ciclo de desenvolvimento, foi solicitada a primeira versão da ferramenta para realizar um alinhamento simples de dados. A tarefa consistia em alinhar os dados com base no campo do CPF, requerendo que o alinhamento fosse feito de forma precisa e exata. Essa funcionalidade foi priorizada para atender a uma demanda específica de alinhamento de dados com base nesse campo-chave.

## 3.2 Segundo ciclo

No segundo ciclo de desenvolvimento, o desafio proposto foi o alinhamento aproximado entre pessoas e endereços. Como muitas vezes as pessoas inserem seus endereços de forma não padronizada, era necessário lidar com essa variação nos dados.

Para atender a esse requisito, foram adicionados campos de aproximação e foram implementados valores para a técnica de sinônimos. Isso permitiu que a ferramenta fosse capaz de identificar correspondências aproximadas entre os dados de pessoas e endereços, levando em consideração as variações e possíveis sinônimos utilizados. Essa melhoria no processo de alinhamento contribuiu para uma maior precisão e abrangência na integração dos dados.

## 3.3 Terceiro ciclo

No terceiro ciclo de desenvolvimento, o desafio proposto foi a união de múltiplas tabelas com base em um campo de nomes. Era necessário integrar as informações dessas tabelas, levando em conta a variação nos nomes e a possibilidade de diferentes formatações.

Para enfrentar esse desafio, foi adicionada uma funcionalidade de quebra de indivíduos, permitindo ao usuário definir um caractere ou grupo de caracteres como ponto de separação dos indivíduos. Isso possibilitou a integração das informações das tabelas, mesmo quando os nomes estavam apresentados de forma diferente em cada uma delas.

Essa funcionalidade de quebra de indivíduos contribuiu para uma maior flexibilidade na união das tabelas, permitindo que a ferramenta realizasse a integração com base nas preferências e necessidades do usuário, resultando em uma integração mais precisa e abrangente dos dados.

## 3.4 Desafios

Durante a parceria estabelecida com a Prefeitura para o desenvolvimento desta dissertação, foram constatadas diversas dificuldades, que podem ser resumidas nas seguintes observações:

- Dificuldade na utilização da ferramenta via linha de comando: A ferramenta exigia que o usuário seguisse vários passos para utilizá-la corretamente, desde a colocação dos arquivos nos formatos adequados nas pastas corretas até a escolha das melhores opções para cada caso. Essa complexidade dificultava o uso da ferramenta de forma eficiente e intuitiva.

- Dificuldade na instalação do projeto: Devido à ampla variedade de bibliotecas utilizadas no projeto, a instalação da ferramenta podia ser problemática. Conflitos

entre as bibliotecas e problemas de incompatibilidade poderiam ocorrer, resultando em mau funcionamento da ferramenta ou até mesmo em falhas na instalação.

- Dificuldade com o formato dos dados: No início do desenvolvimento, foram realizadas integrações com diversos formatos de dados, como SQL, *CSV* e NoSQL. No entanto, lidar com a variabilidade de como os dados estavam armazenados em cada formato de arquivo provou ser um desafio. Diferenças na estrutura e organização dos dados resultavam em erros no processo de alinhamento.

Esses desafios foram fundamentais para direcionar o desenvolvimento da ferramenta e resultaram na criação de uma interface mais amigável. O objetivo principal era simplificar a instalação e orientar os usuários sobre o uso adequado da ferramenta. Como parte desse processo, foram adotadas algumas medidas, as quais estão descritas a seguir:

Simplificação da abordagem inicial dos arquivos: foi decidido por aceitar apenas arquivos com a extensão *CSV*, a fim de simplificar o processo de tratamento e alinhamento dos dados. Essa restrição permitiu que o foco da dissertação fosse direcionado a padronização e processamento eficiente dos arquivos.

Criação de um Makefile: Foi Desenvolvido um arquivo Makefile que simplifica o processo de instalação da ferramenta. Com um único comando, os usuários podem configurar e instalar todas as dependências necessárias, tornando o processo mais rápido e eficiente.

Essas medidas foram implementadas para aprimorar a usabilidade e a experiência geral do usuário, tornando a instalação da ferramenta mais fácil e direta. A seguir, a arquitetura em que a ferramenta foi baseada será apresentada.

## 3.5 Arquitetura

A arquitetura foi criada em paralelo ao desenvolvimento de uma ferramenta de validação. O propósito da arquitetura é executar quatro etapas distintas: conversão, equivalência de campos, equivalência de propriedades e geração de resultados. O fluxo dessas etapas é ilustrado na Figura 5, proporcionando uma representação visual do funcionamento da arquitetura.

A arquitetura foi meticulosamente concebida com a fundação nos princípios dos microsserviços, com o propósito de permitir que cada usuário execute alinhamentos de dados de forma personalizada e alcance resultados satisfatórios para uma ampla variedade de

conjuntos de dados. Além disso, essa arquitetura viabiliza a inclusão de novos serviços, garantindo uma ferramenta expansível e adaptável para atender às necessidades específicas de cada caso.

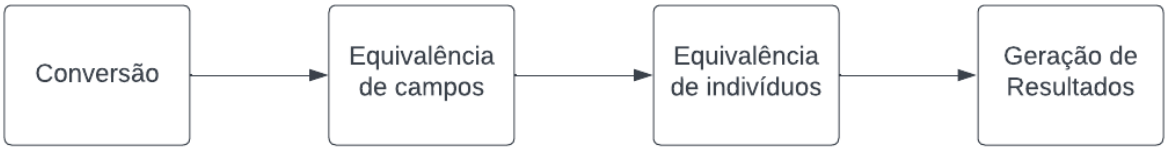


Figura 5: Etapas da arquitetura

### 3.5.1 Conversão

A etapa de conversão é composta por quatro etapas-chave: recebimento dos dados a serem alinhados, agrupamento de dados previamente conhecidos pelo usuário, seleção de serviços a serem utilizados e formatação das informações, quando necessário. Essas etapas têm como objetivo alcançar resultados mais precisos no alinhamento dos dados e permitem a utilização de múltiplos tipos de dados.

No processo de recebimento dos dados, o usuário é requisitado a fornecer os dados a serem alinhados, os quais podem apresentar uma diversidade de formatos e diferentes aridades em relação aos seus campos. Por exemplo, é possível realizar o alinhamento entre duas bases, onde a primeira pode conter os campos “nome” e “cpf”, enquanto a segunda base pode ter apenas o campo “cpf”. Essa flexibilidade possibilita o tratamento de uma ampla gama de dados de entrada, independentemente do formato ou estrutura em que se encontram.

O passo de agrupamento de dados previamente conhecidos pelo usuário tem como objetivo agregar o máximo de informações possíveis sobre os dados. Por exemplo, ao alinhar duas bases de dados sobre pessoas, o usuário pode optar por focar o alinhamento nos campos “nome” de uma base e “nome completo” de outra base. Dessa forma, o alinhamento será realizado com base nos campos selecionados pelo usuário, permitindo uma correspondência mais precisa entre os dados.

No passo de seleção de serviços, o usuário pode escolher os microsserviços considerados necessários para obter o melhor alinhamento dos dados. Além disso, deve ser incluído um microsserviço padrão que será executado caso nenhum outro seja selecionado. Essa flexibilidade permite personalizar o processo de alinhamento de acordo com as necessidades

específicas do usuário.

No passo de formatação dos dados, é considerado que as bases de dados fornecidas pelo usuário podem não estar formatadas corretamente ou podem exigir modificações em seus indivíduos sob diferentes perspectivas. Além disso, pode ser necessário formatar os dados de maneiras diferentes para cada tipo de base de dados. Nessa etapa, é solicitado ao usuário que indique se os dados devem ser reestruturados e permite-se receber novos módulos para-se atuar nos diferentes formatos das bases.

Essas etapas da conversão são fundamentais para obter resultados de alinhamento mais precisos, permitindo lidar com diversos tipos de dados e personalizar o processo de alinhamento de acordo com as preferências e requisitos do usuário.

### 3.5.2 Equivalência de campos

A etapa de equivalência de campos ocorre após a formatação das bases de dados fornecidas pelo usuário e a definição das parametrizações. Nessa fase, os microsserviços selecionados pelo usuário são executados em uma ordem predefinida. Uma característica crucial desses serviços é a sua independência, garantindo que a execução de um serviço não tenha impacto sobre o funcionamento dos demais. Além disso, cada serviço deve gerar um resultado que pode ser usado como tabela final ou ser processado automaticamente pelos outros serviços. A facilidade para adicionar novos microsserviços à ferramenta é fundamental, permitindo que o usuário personalize seus processos de alinhamento de forma simples e flexível.

Cada serviço, por sua vez, examina cada campo das bases de dados e tenta identificar quais destes campos são equivalentes de acordo com as validações específicas do serviço. Por exemplo, um serviço que verifica igualdades consideraria que o campo A é equivalente ao campo B se eles forem iguais. Já um serviço que avalia a distância entre o texto do campo A e o campo B consideraria os campos equivalentes se passassem por uma taxa de aceitação definida pelo serviço. Esse processo de alinhamento é realizado para todos os campos definidos pelo usuário em cada base de dados fornecida. Caso o usuário não selecione campos específicos para serem verificados, o serviço deve examinar todos os campos de todas as bases.

Dessa forma, a etapa de equivalência de campos realiza a comparação entre os campos das bases de dados, buscando identificar as equivalências conforme as validações definidas pelos serviços selecionados pelo usuário. Essa abordagem permite flexibilidade na defi-

nição dos critérios de equivalência e garante que todas as combinações de campos sejam consideradas, caso o usuário não especifique campos específicos.

### 3.5.3 Equivalência de propriedades

A etapa de equivalência de propriedades ocorre após o alinhamento dos campos nas bases de dados fornecidas, realizado na etapa anterior, e a definição das parametrizações pelo usuário. Nessa etapa, os microsserviços selecionados pelo usuário são executados seguindo a mesma lógica da etapa anterior.

Cada serviço utiliza os campos definidos como equivalentes na etapa anterior para realizar as verificações nos valores dos indivíduos correspondentes. Por exemplo, se houver uma equivalência entre os campos "nome" e "nome completo" em duas bases de dados, a verificação das propriedades ocorrerá apenas nos valores desses campos, ignorando os demais valores que os indivíduos possam conter. Para que dois indivíduos sejam considerados equivalentes, todos os valores dos campos designados como equivalentes devem ser também considerados equivalentes.

Por exemplo, considerando três bases de dados distintas, em que a base A teve equivalência com a base B para o campo "nome" e equivalência com a base C para os campos "nome" e "idade". Nesse caso, uma equivalência entre um indivíduo da base A e B será válida quando os valores de "nome" entre os dois indivíduos forem considerados equivalentes pelo serviço selecionado. Já para que a equivalência seja válida entre os indivíduos das bases A e C, os indivíduos devem possuir os valores de "nome" e "idade" equivalentes. Assim, caso apenas o valor de "nome" ou de "idade" seja equivalente, os indivíduos não serão considerados equivalentes entre as bases.

Essa abordagem permite realizar a verificação das propriedades com base nos valores dos campos previamente alinhados, garantindo que os indivíduos sejam considerados equivalentes somente se todos os valores correspondentes dos campos definidos como equivalentes também forem equivalentes.

### 3.5.4 Geração de Resultados

A etapa de geração de resultados é dividida em três objetivos principais: gerar arquivos alinhados para o usuário, possibilitar inferências e melhorias contínuas nos dados alinhados, e estabelecer uma conexão com a ontologia SUMO. Esses objetivos visam proporcionar melhorias contínuas nos dados fornecidos.

No passo de geração dos arquivos alinhados, são gerados arquivos que podem ser utilizados para inferências futuras e para realizar novos alinhamentos de dados. No passo de novas inferências, a ferramenta desenvolvida deve ser capaz de realizar automaticamente inferências com base na estrutura definida durante sua construção. Além disso, deve ser possível passar por todo o processo de alinhamento novamente, com ou sem a inclusão de novas bases de dados, permitindo a melhoria contínua do alinhamento.

No passo de conexão com a ontologia SUMO, os campos dos dados alinhados são verificados em relação aos dados da Wordnet. Para cada termo presente tanto na Wordnet quanto nos dados alinhados, é estabelecida uma ligação com os elementos correspondentes na SUMO, utilizando os IDs da Wordnet e os dados da SUMO. Essa ligação envolve a criação de uma nova classe de dados na ontologia e o estabelecimento de relações entre os conceitos equivalentes. Caso algum campo não possua uma ligação com algum termo da Wordnet, sua adição é feita na ontologia sem relacionamentos. Cada indivíduo terá seus valores separados para os campos e adicionados dentro da nova classe de dados criada, com uma relação para denotar o indivíduo. Para esse objetivo, é aplicado um grupo de raciocinadores chamados TPTP para realizar operações na ontologia.

Esses três focos da etapa de geração de resultados permitem gerar arquivos alinhados, realizar inferências e melhorias contínuas nos dados alinhados, além de estabelecer uma conexão com a ontologia SUMO para enriquecer a representação dos dados e possibilitar análises mais avançadas. A seguir, o algoritmo desenvolvido será apresentado.

## 3.6 Algoritmo

A implementação deste projeto foi desenvolvida utilizando Python 3.11.0 em conjunto com o pip 22.3. O código-fonte do projeto está disponível no seguinte repositório do GitHub:
https://github.com/frame-lab/interoperaNit.

A versão atual da ferramenta foi desenvolvida com base nas versões dos pacotes listadas a seguir. Para garantir o pleno funcionamento da ferramenta, é essencial que as dependências estejam devidamente instaladas, sendo aconselhável seguir as versões especificadas abaixo.

- deepmatcher: 0.1.2.post2
- torchtext: 0.9.0

- torch: 1.8.0
- textdistance: 4.2.2
- nltk: 3.6.7
- python-dotenv: 0.19.2
- google-cloud-translate: 3.6.1
- pandas: 1.4.0
- PyQt5: 5.15.6
- py-entitymatching: 0.4.0
- tk: 0.1.0
- numpy: 1.22.1
- pandasql: 0.7.3
- tqdm: 4.62.3

Embora não seja obrigatório utilizar as versões específicas dos pacotes e as versões do Python e do pip mencionadas, elas foram testadas e comprovam o correto funcionamento da ferramenta. Isso significa que versões diferentes podem ser utilizadas, mas é recomendado utilizar as versões testadas para garantir a melhor compatibilidade e desempenho.

Uma opção simplificada de instalação também está disponível, utilizando os comandos "make install_python" para instalar a versão recomendada do Python e adicioná-la às variáveis de ambiente do sistema. Além disso, o comando "make install_programs" permite instalar os programas necessários para o funcionamento geral da ferramenta, que estão listadas a seguir.

- unzip
- git
- gcc
- graphviz
- ant

- pip

Caso o computador onde a ferramenta esteja sendo utilizada já possua esses programas, é possível pular a etapa do "make install_programs" e avançar para a instalação dos pacotes e ferramentas listados acima utilizando o comando "make install". Além de realizar essas instalações, esse comando também gera a estrutura de arquivos necessária para a ferramenta, conforme listado a seguir.

**samples:** Pasta que contém as bases de entrada para o programa.

**results:** Pasta onde serão armazenados os arquivos de consultas gerados.

**csv:** Pasta onde será gerado o arquivo contendo a tabela unificada completa.

**unique:** Arquivo que define as palavras que devem ser consideradas únicas durante o processo de alinhamento, como CPF e Email.

**queries:** Arquivo que define as consultas a serem executadas ao final do processamento, utilizando operadores lógicos.

**approximate:** Arquivo que define os campos que podem ter suas avaliações de propriedades realizadas por meio de algoritmos de aproximação.

**split:** O arquivo de configuração "split" define os campos e caracteres (no formato "campo" "caractere") que resultarão na duplicação de linhas. Cada ocorrência do "caractere" na coluna "campo" irá gerar uma nova linha, separando o conteúdo à esquerda do "caractere" na primeira linha e o conteúdo à direita na linha seguinte. Essa duplicação de linhas resultará em múltiplos indivíduos, mantendo todos os outros campos inalterados.

A estrutura de arquivos e pastas também pode ser criada manualmente, adicionando esses arquivos e pastas à pasta raiz da ferramenta, chamada "interopera". Com o ambiente configurado, o projeto poderá ser executado e seguirá o fluxo ilustrado na Figura 6. Esse fluxo representa as etapas e a sequência de ações a serem realizadas durante a execução da ferramenta.

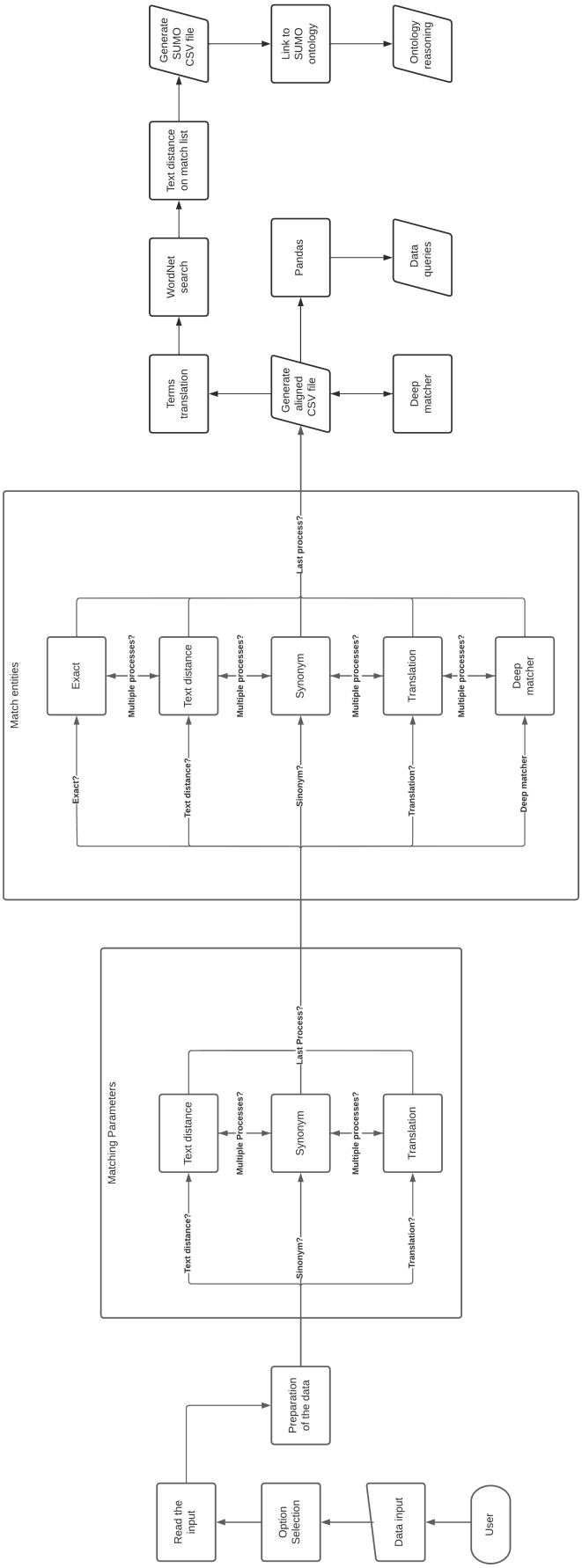


Figura 6: Fluxo da ferramenta

Após configurar o ambiente necessário, a ferramenta pode ser executada ao executar o comando "python controler -args", em que "-args" deve ser substituído por uma ou várias opções dos comandos listados na Descrição 3.6. Em seguida, a etapa de conversão da ferramenta começará, seguindo o modelo da arquitetura, conforme descrito na Seção 3.6.1.

**-h:** Mostra o arquivo de ajuda.

**-e <percent>:** Executa o algoritmo utilizando o método de distância de edição, onde a taxa de aprovação para dois elementos serem considerados iguais é definida como *percent*.

**-t:** Executa o algoritmo utilizando o método de tradução, com o idioma inglês como destino da tradução.

**-s:** Executa o algoritmo utilizando o método de sinônimos.

**-dm:** Executa o algoritmo utilizando o método de Deep Matcher.

**-a:** Executa o algoritmo com aproximação para o alinhamento dos campos.

**-v:** Executa o algoritmo registrando o processo de alinhamento passo a passo.

**-val:** Executa o algoritmo com uma validação dos arquivos de entrada para verificar se estão corretos.

**-max:** Executa o algoritmo com aproximação, onde os elementos serão considerados correspondentes com base na maior taxa de aproximação possível. Funciona apenas com o método de distância de edição.

### 3.6.1 Conversão

A etapa de conversão tem início com a leitura dos arquivos, a menos que o algoritmo seja executado com o comando "-h". Nesse caso, ele exibirá primeiro o conteúdo do arquivo de ajuda, conforme mostrado no *Listing* 3.1. Em seguida, o algoritmo dará continuidade à execução, voltando para a leitura dos arquivos. Além disso, é possível realizar essa execução junto ao comando "-v", o qual mostrará cada etapa executada e o respectivo passo realizado.

```
1 Interopera - Makes an alingment between files in the CSV format
2 Usage: python controler -args
3 -h                Prints this help
```

```
4  -e <percent>     Executes the algorithm to make an align of the bases
5  with editing distance
6  -t               Executes the algorithm to make an align of the bases
7  with translation
8  -s               Executes the algorithm to make an align of the bases
9  with synonyms
10 -dm              Executes the algorithm to make an align of the
11 entities with Deep matcher
12 -a               Executes the algorithm with approximate matches in
       the
13 align of the entities
14 -v               Executes the algorithm with the verbose option active
15 -val             Executes the algorithm with a validation of the files
16 in the beginning
17 -max             Executes the algorithm with the maximum approximation
18 of the results
```

Listing 3.1: Arquivo de ajuda

No momento, a ferramenta aceita apenas arquivos no formato *CSV* para a leitura. No entanto, a adição de novos formatos pode ser facilmente realizada sem afetar o restante da ferramenta ou a forma como os arquivos *CSV* são lidos. Basta adicionar um novo módulo de análise para o formato desejado. Quando esses módulos forem adicionados, o programa executará o alinhamento em todos os formatos aceitos, uma vez que o módulo criará um objeto comum para o alinhamento, como ilustrado na *Listing* 3.2.

```
1  {
2      name: Nome da tabela
3      parameters: Lista de campos
4      entities: Lista de indivíduos
5      extension: Formato da extensão do arquivo original
6      file_name: Nome do arquivo
7      match_name: Lista de emparelhamentos das bases
8      match_parameters: Lista de emparelhamentos de campos
9      match_entities: Lista de emparelhamentos de indivíduos
10 }
```

Listing 3.2: Objeto de base

Para incorporar um novo módulo de análise, o processo envolve a criação de um

arquivo Python na pasta “parsers” localizada na pasta “src” dentro da raiz do projeto. Este arquivo deve ser configurado para receber uma lista de linhas a serem processadas, uma lista de caracteres para delimitar a quebra dos indivíduos, a variável booleana de execução em modo verboso, a lista de campos a serem executados de forma aproximada, a variável booleana de se aproximar todos os campos e uma lista de palavras a serem consideradas únicas.

Os dados devem ser processados com base nas informações fornecidas ao arquivo, e após a conclusão do processamento, o arquivo de análise deve disponibilizar os dados relativos aos campos e indivíduos identificados.

Uma vez que o arquivo de análise esteja pronto, é necessário importá-lo no arquivo “*preparer*” localizado na pasta "*src*"e acrescentar uma nova opção de extensão à função “*prepare_bases*” de acordo com o formato desejado. Além disso, os dados precisam ser transmitidos para o arquivo de análise. Por fim, esses dados devem ser usados para criar o objeto de base correspondente, em conjunto com as informações presentes no arquivo.

Para ilustrar a geração do objeto de base, o *Listing* 3.3 representa um arquivo *CSV* chamado “Pessoa.csv”. Esse arquivo será processado para criar o objeto base da ferramenta.

```
1  Name,Docs
2  Caio,74051264051a
3  William,62154377603
4  Silva,42587155062
5  Patricia,45109110094
6  Bruna,32282832087
7  Alice,76401849089
8  Jorge,67390268078
9  Flavia,25095837007
10 Tatiane,13545955055
11 Paulo,01131789057
```

Listing 3.3: Exemplo de arquivo csv

A criação do objeto base ocorre através da passagem dos campos *name*, *extension* e *file_name*, que representam o nome da tabela, o formato do arquivo e o nome do arquivo, respectivamente. Em seguida, a primeira linha do arquivo *CSV* é processada, e todos os campos encontrados são adicionados ao campo *parameters*. Posteriormente, cada linha subsequente é processada como um indivíduo, sendo adicionada ao campo *entities*. O

*Listing* 3.4 ilustra um exemplo desse resultado.

```
{
    name: Pessoa
    parameters: [Name, Docs]
    entities: [[Caio,74051264051a], [William,62154377603], [Silva
        ,42587155062]]
    extension: csv
    file_name: Pessoa
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.4: Objeto de base para Pessoa

Existe um processo alternativo de criação do objeto base que ocorre quando o usuário define palavras de quebra de linha no arquivo *split*. Nesse cenário, o objeto base é criado da mesma forma que no processo normal até a etapa de construção da lista de indivíduos. Quando o caractere definido como quebra de linha é encontrado, o indivíduo é dividido em n+1 partes, sendo n a quantidade de vezes que o caractere aparece. No entanto, uma parte não poderá ser vazia e, portanto, é excluída da lista. Ao quebrar uma linha, todos os outros campos do indivíduo são mantidos. Dado o arquivo de entrada fornecido em *Listing* 3.5, e o arquivo de quebra de linha fornecido em *Listing* 3.6. A criação do objeto pode ser observada no *Listing* 3.7.

```
Names,Docs
Caio/Carlos,74051264051a
William/Weston/,62154377603
Silva,42587155062
Patricia,45109110094
Bruna,32282832087
Alice,76401849089
Jorge,67390268078
Flavia,25095837007
Tatiane,13545955055
Paulo,01131789057
```

Listing 3.5: Exemplo de arquivo csv

```
Names /
```

Listing 3.6: Exemplo de arquivo de quebra de linha

```
{
    name: Pessoa
    parameters: [Name, Docs]
    entities: [[Caio,74051264051a], [Carlos,74051264051a], [William
        ,62154377603], [Weston,62154377603], [Silva,42587155062]]
    extension: csv
    file_name: Pessoa
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.7: Objeto de base para Pessoa

Neste exemplo, a quebra de linha ocorre sempre que o caractere "/" é encontrado na coluna "*names*". Por exemplo, o indivíduo "Caio/Carlos,74051264051a" será dividido em dois indivíduos: "Caio,74051264051a" e "Carlos,74051264051a". Além disso, o indivíduo "William/Weston/,62154377603" será dividido em três indivíduos: "William,62154377603", "Weston,62154377603" e ",62154377603". No entanto, esse último indivíduo será excluído da lista, pois o campo "*names*" ficou vazio após a quebra de linha.

Caso o usuário não esteja confiante de que os dados de entrada estejam corretos, é possível executar o algoritmo com um processo de validação das entradas. Para isso, o comando -val será utilizado durante a execução do algoritmo, o que processará cada arquivo e, caso haja algum erro, exibirá uma mensagem de erro para o usuário. Por exemplo, considerando a entrada "Pessoa.csv" presente no *Listing* 3.8, o retorno de erro resultante da execução do algoritmo de validação pode ser encontrado no *Listing* 3.9.

```
Names,Docs
Caio,74051264051a,erro
William,62154377603
```

Listing 3.8: Exemplo de arquivo csv com erro

```
An error of column was found in the line 2 of the file Pessoa:
Number of columns is 3 when it should be 2.
```

Listing 3.9: Exemplo de saida de erro

Neste exemplo, o algoritmo de validação identificou que há uma coluna a mais presente na segunda linha do arquivo de entrada. Em resposta a essa inconsistência, o algoritmo abortaria o processo de execução padrão, a fim de evitar erros e garantir que os dados estejam corretos antes de prosseguir e mostraria a mensagem de erro do *Listing* 3.9.

Após a conclusão da leitura e processamento dos arquivos, o algoritmo prosseguirá com o alinhamento dos campos encontrados nas bases, conforme descrito pela Seção 3.6.2. Esse passo será executado com o algoritmo de alinhamento exato, a menos que o usuário tenha fornecido o argumento -a junto com alguma das opções de alinhamento, como "-e", "-t", "-s" ou "-dm". Se o usuário desejar executar o alinhamento aproximado apenas para uma parte dos campos, ele deverá, em vez de usar o comando -a, adicionar o campo à lista de campos do arquivo "approximate". Contudo mesmo que o usuário selecione um destes métodos o alinhamento pelo método exato continuara será executado.

### 3.6.2 Equivalência de campos

A primeira etapa do alinhamento consiste em verificar se o usuário adicionou algum campo no arquivo *unique*. Caso algum dos campos tenha sido adicionado, apenas esses campos serão considerados para o alinhamento, desde que também sejam encontrados nas tabelas. No entanto, se uma tabela não possuir um campo *unique*, todos os campos serão utilizados no alinhamento. O algoritmo irá então realizar uma combinação de todos os campos necessários, utilizando as técnicas selecionadas pelo usuário, e caso alguma das técnicas detecte que houve um emparelhamento, a combinação também será considerada um emparelhamento. Por fim os dados do emparelhamento são salvos na primeira tabela que está sendo realizada a comparação, o objeto resultante do emparelhamento pode ser visto no *Listing* 3.10.

```
1 {
2     name: "Nome da base a qual o emparelhamento foi encontrado",
3     matched_parameter: "Nome do campo da base que ocorreu o
          emparelhamento",
4     my_parameter: "Nome do campo que ocorreu emparelhamento",
5     approximate: "Se o emparelhamento ocorreu de forma aproximada ou
          não"
6 }
```

Listing 3.10: Objeto de emparelhamento de campos

Para adicionar novos serviços à ferramenta, deve ser seguido o procedimento: um novo

arquivo Python deve ser criado na pasta “techniques”, localizada dentro da pasta “src” na raiz do projeto. Este arquivo deve receber duas propriedades ou, no caso de um algoritmo baseado em inteligência artificial, as duas bases de dados que serão comparadas.

Em seguida, as funções relacionadas à execução do serviço gerado devem ser adicionadas no arquivo “aligner"encontrado na pasta “src” No arquivo “aligner” os usuários devem chamar as funções “align_entities” para o alinhamento de indivíduos, especificando a referência da função a ser chamada, o tipo de comparação, se necessário, e a indicação de se os dados devem ser aproximados. Além disso, a função “align_bases"deve ser referenciada para o alinhamento dos campos, especificando a referência da função a ser chamada.

Para algoritmos baseados em inteligência artificial, a função “align_entities_ia"deve ser utilizada em vez da função “align_entities”, especificando a referência da função a ser chamada.

Por fim, novas funções devem ser adicionadas ao arquivo “controller” para chamar o novo serviço, definindo um novo argumento que faça referência ao serviço recém-criado e incluindo o serviço na lista de funções a serem executadas, caso o argumento tenha sido selecionado. Esse procedimento oferece a flexibilidade de personalizar a ferramenta ao adicionar novos serviços de acordo com as necessidades específicas.

A complexidade do algoritmo abordado nesta dissertação está intrinsecamente ligada aos módulos de analise e serviços que serão empregados no processo. É importante notar que quanto mais serviços forem selecionados, maior será o tempo computacional necessário para realizar o alinhamento dos dados.

Um exemplo ilustrativo de como esse alinhamento dos campos ocorre pode ser observado nos Exemplos 3.6.2.1, 3.6.2.2 e 3.6.2.3. Cada exemplo demonstra o processo de verificação e correspondência dos campos, apresentando casos em que o alinhamento resulta em correspondências exatas e em casos em que a correspondência é obtida por meio dos outros método.

#### 3.6.2.1 Exemplo de alinhamento dos campos

Neste exemplo o objetivo é alinhar os campos de duas tabelas de pessoa descritas nos *Listings* 3.11 e 3.13 focando apenas nos campos “*Names*” e “*Name*”, para isso o arquivo *unique* será preenchido como no *Listing* 3.15 e as representações dos objetos dessas tabelas estão descritos pelos *Listings* 3.12 e 3.14 respectivamente. A Figura 7 ilustra o fluxograma das etapas de alinhamento e seus respectivos processos.

```
Names,Docs,birth
```

Listing 3.11: Exemplo de entrada de pessoa 1

```
{
    name: Pessoa1
    parameters: [Names,Docs,birth]
    entities: []
    extension: csv
    file_name: Pessoa1
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.12: Objeto de base para Pessoa 1

```
Name,address,email
```

Listing 3.13: Exemplo de entrada de pessoa 2

```
{
    name: Pessoa2
    parameters: [Name,address,email]
    entities: []
    extension: csv
    file_name: Pessoa2
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.14: Objeto de base para Pessoa 2

```
Name
Names
```

Listing 3.15: Exemplo de arquivo unique

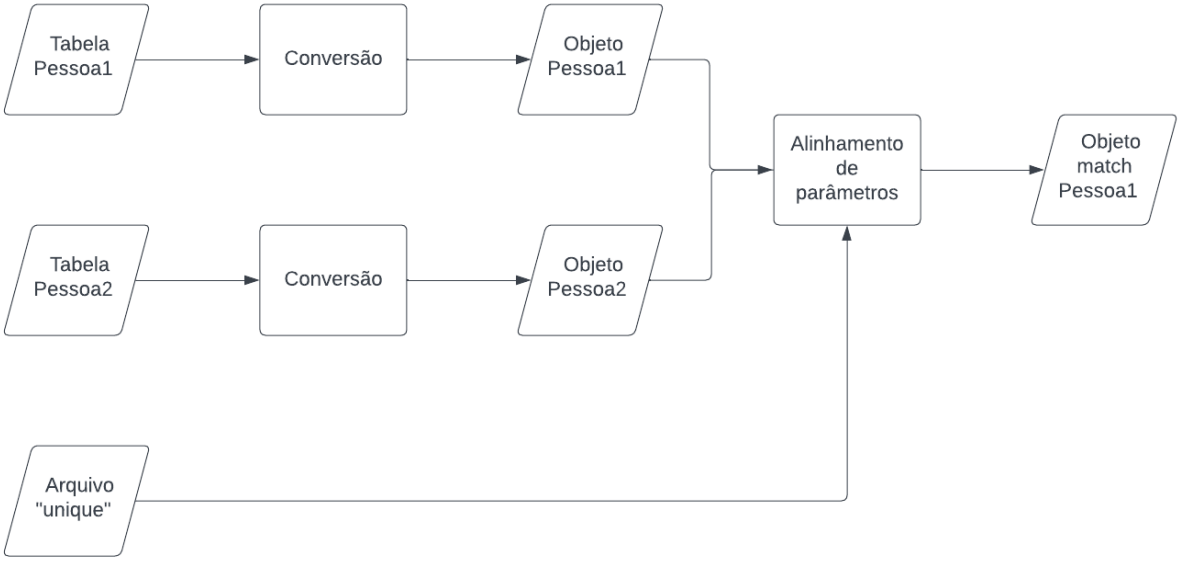


Figura 7: Fluxograma do alinhamento de pessoas.

O programa deverá ser executado utilizando-se o comando "-a". Isso é necessário porque existe uma discrepância entre os campos selecionados como *unique*, o que acarreta na utilização das técnicas não exatas para o alinhamento dos campos. Devido à pequena diferença de apenas uma letra entre os campos, um algoritmo eficiente para esse propósito seria o de distância de edição.

Para este caso o alinhamento irá verificar apenas os campos *unique* de cada uma das tabelas visto que "*Names*" está presente na Tabela 3.11 e "*Name*" está presente na Tabela 3.13, ele então verificara se estes campos serão um emparelhamento pelo método exato o qual resultara em um resultado falso e em sequencia verificara se pelo método da distancia de edição o qual resultara em um resultado verdadeiro, marcando o resultado como emparelhamento no objeto da Tabela 3.12 resultando no objeto descrito no *Listing* 3.16

```
1 {
2     name: Pessoa1
3     parameters: [Names,Docs,birth]
4     entities: []
5     extension: csv
6     file_name: Pessoa1
7     match_name: []
8     match_parameters: [
9         {
```

```
10                name: Pessoa2,
11                matched_parameter: Name,
12                my_parameter: Names,
13                approximate': true
14            }
15        ]
16        match_entities: []
17    }
```

Listing 3.16: Objeto de base para Pessoa 1

#### 3.6.2.2 Exemplo de alinhamento dos campos

Neste exemplo o objetivo é alinhar os campos de duas tabelas de Dna descritas nos *Listings* 3.17 e 3.19 focando em todos os campos disponíveis para isto um arquivo "*unique*" não será necessário e as representações dos objetos dessas tabelas estão descritos pelos *Listings* 3.18 e 3.20 respectivamente. A Figura 8 ilustra o fluxograma das etapas de alinhamento e seus respectivos processos.

```
1  sample_id,location,sample_type
```

Listing 3.17: Exemplo de entrada de Dna 1

```
1  {
2      name: Dna1
3      parameters: [sample_id,location,sample_type]
4      entities: []
5      extension: csv
6      file_name: Dna1
7      match_name: []
8      match_parameters: []
9      match_entities: []
10 }
```

Listing 3.18: Objeto de base para Dna 1

```
1  sample_id,localização,sample_type
```

Listing 3.19: Exemplo de entrada de Dna 2

```
{
    name: Dna2
    parameters: [sample_id,localização,sample_type]
    entities: []
    extension: csv
    file_name: Dna2
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.20: Objeto de base para Dna 2

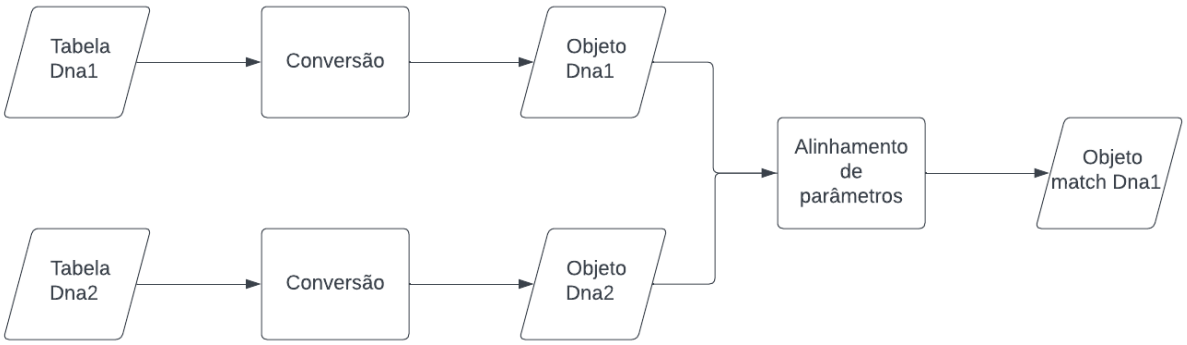


Figura 8: Fluxograma do alinhamento de Dnas.

O programa deverá ser executado utilizando-se o comando "-a". Isso é necessário porque existe uma discrepância entre os campos "location" e "localização", o que acarreta na utilização das técnicas não exatas para o alinhamento dos campos. Devido à diferença de idioma entre os campos, um algoritmo eficiente para esse propósito seria o de tradução.

Neste caso, o alinhamento verificará todos os campos, analisando inicialmente se eles correspondem exatamente, o que resultará em um resultado verdadeiro para "sample_id" e "sample_type", uma vez que esses possuem o mesmo valor em ambas as tabelas. Porém, resultará em um resultado falso para a correspondência entre "location" e "localização". Em seguida, o alinhamento verificará se há uma correspondência por meio do método de tradução, o qual resultará em um resultado verdadeiro para "location" e "localização", mas em um resultado falso para "sample_id" e "sample_type". O resultado será então marcado como um emparelhamento no objeto da Tabela 3.18, resultando no objeto descrito no

*Listing* 3.21.

```
{
    name: Dna1
    parameters: [sample_id,location,sample_type]
    entities: []
    extension: csv
    file_name: Dna1
    match_name: []
    match_parameters: [
        {
            name: Dna2,
            matched_parameter: sample_id,
            my_parameter: sample_id,
            approximate': false
        },
        {
            name: Dna2,
            matched_parameter: localização,
            my_parameter: location,
            approximate': true
        },
        {
            name: Dna2,
            matched_parameter: sample_type,
            my_parameter: sample_type,
            approximate': false
        }
    ]
    match_entities: []
}
```

Listing 3.21: Objeto de base para Dna 1

#### 3.6.2.3 Exemplo de alinhamento dos campos

Neste exemplo, o objetivo é alinhar os campos de três tabelas de Cidades, descritas nos *Listings* 3.22, 3.24 e 3.26, focando nos campos de "Cidade" e "País". Para isso, o arquivo "unique" será preenchido conforme o exemplo do *Listing* 3.28, e as representações dos

objetos dessas tabelas estão descritas nos *Listings* 3.23, 3.25 e 3.27, respectivamente. A Figura 9 ilustra o fluxograma das etapas de alinhamento e seus respectivos processos.

```
Escritório,Cidade,País,Capacidade
```

Listing 3.22: Exemplo de entrada de Cidade 1

```
{
    name: Cidade1
    parameters: [Escritório,Cidade,País,Capacidade]
    entities: []
    extension: csv
    file_name: Cidade1
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.23: Objeto de base para Cidade 1

```
Filial,Cidade,País,Receita Anual
```

Listing 3.24: Exemplo de entrada de Cidade 2

```
{
    name: Cidade2
    parameters: [Filial,Cidade,País,Receita Anual]
    entities: []
    extension: csv
    file_name: Cidade2
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.25: Objeto de base para Cidade 2

```
Ponto Turístico,Cidade,País,Visitantes Anuais
```

Listing 3.26: Exemplo de entrada de Cidade 3

```
{
    name: Cidade3
```

```
3      parameters: [Ponto Turístico,Cidade,País,Visitantes Anuais]
4      entities: []
5      extension: csv
6      file_name: Cidade3
7      match_name: []
8      match_parameters: []
9      match_entities: []
10 }
```

Listing 3.27: Objeto de base para Cidade 3

```
1 Cidade
2 País
```

Listing 3.28: Exemplo de arquivo unique

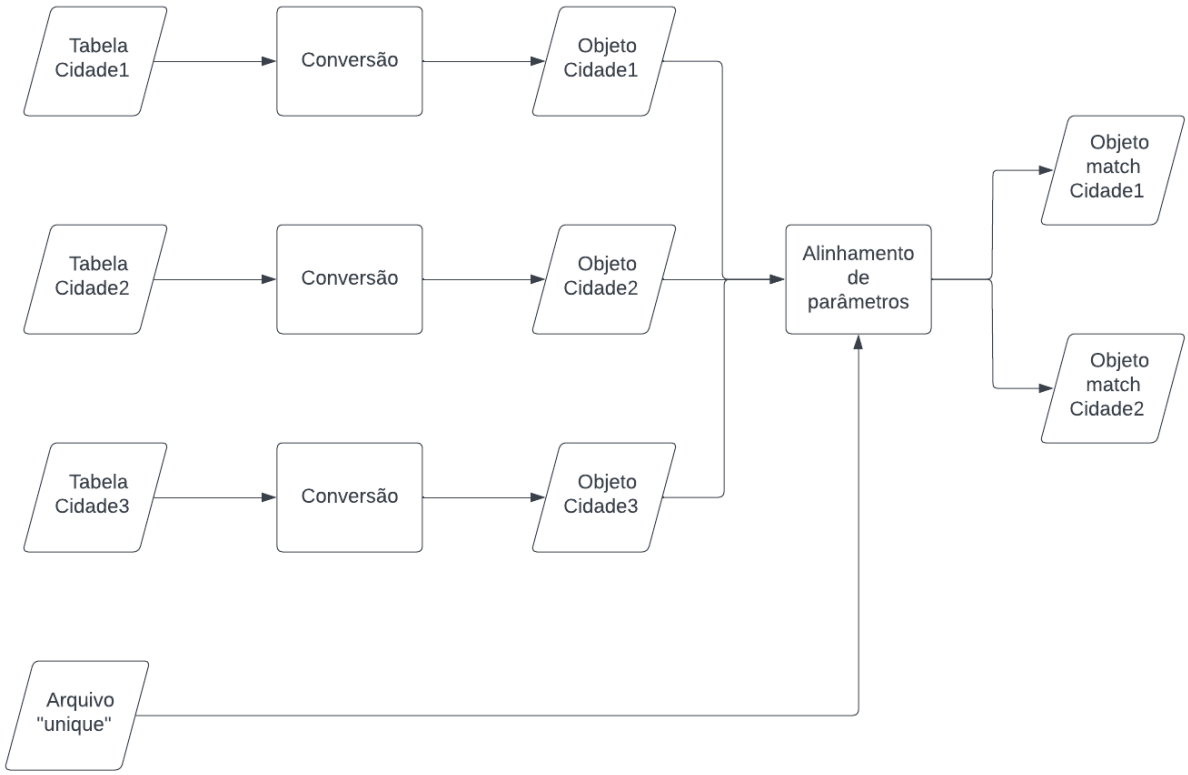


Figura 9: Fluxograma do alinhamento de cidades.

Para este caso, o programa pode ser executado sem a utilização do comando "-a", uma vez que os campos a serem alinhados possuem os mesmos valores. O programa verificará os campos presentes no arquivo "unique" de cada uma das tabelas e, em seguida, verificará

se esses campos são iguais pelo método exato. Como resultado, os campos entre as três tabelas serão considerados correspondentes, e o algoritmo marcará os Objetos 3.23 e 3.25 como correspondentes, resultando nos objetos descritos nos *Listings* 3.29 e 3.30.

```
1  {
2      name: Cidade1
3      parameters: [Escritório,Cidade,País,Capacidade]
4      entities: []
5      extension: csv
6      file_name: Cidade1
7      match_name: []
8      match_parameters: [
9          {
10             name: Cidade2,
11             matched_parameter: Cidade,
12             my_parameter: Cidade,
13             approximate': false
14         },
15         {
16             name: Cidade2,
17             matched_parameter: País,
18             my_parameter: País,
19             approximate': false
20         },
21         {
22             name: Cidade3,
23             matched_parameter: Cidade,
24             my_parameter: Cidade,
25             approximate': false
26         },
27         {
28             name: Cidade3,
29             matched_parameter: País,
30             my_parameter: País,
31             approximate': false
32         }
33
34     ]
35     match_entities: []
```

```
36 }
```

Listing 3.29: Objeto de base para Cidade 1

```
1  {
2      name: Cidade2
3      parameters: [Filial,Cidade,País,Receita Anual]
4      entities: []
5      extension: csv
6      file_name: Cidade2
7      match_name: []
8      match_parameters: [
9          {
10             name: Cidade3,
11             matched_parameter: Cidade,
12             my_parameter: Cidade,
13             approximate': false
14         },
15         {
16             name: Cidade3,
17             matched_parameter: País,
18             my_parameter: País,
19             approximate': false
20         }
21
22     ]
23     match_entities: []
24 }
```

Listing 3.30: Objeto de base para Cidade 2

### 3.6.3 Equivalência de indivíduos

A etapa de equivalência de indivíduos inicia logo após o término do alinhamento dos campos. O primeiro passo consiste em verificar os campos que foram alinhados e, em seguida, alinhar os indivíduos somente para os valores correspondentes a esses campos. Isso é feito por meio dos serviços de alinhamento selecionados pelo usuário e pelo método exato, combinando todas as propriedades de todos os indivíduos. No entanto, diferentemente do

método de alinhamento de campos, este método considera um emparelhamento entre dois indivíduos somente se eles são um emparelhamento para todos os campos presentes. Por exemplo, consideremos dois indivíduos A e B com emparelhamentos entre os campos CPF e RG; neste caso, o emparelhamento dos indivíduos só será considerado se os valores de CPF e RG para ambos os indivíduos forem um emparelhamento. Ao final deste processo, os dados do emparelhamento são salvos na primeira tabela que realizou a comparação. O objeto resultante do emparelhamento pode ser visualizado no *Listing* 3.31.

```
1 {
2     matched_name: "Nome da base a qual o emparelhamento foi
        encontrado",
3     matched_entity_index: "Index do indivíduo da base que ocorreu o
        emparelhamento",
4     my_parameter_index: "Index do indivíduo que ocorreu o
        emparelhamento",
5 }
```

Listing 3.31: Objeto de emparelhamento de indivíduos

Vários exemplos ilustrativos de como esse alinhamento de indivíduos é realizado podem ser encontrados nas Seções 3.6.3.1, 3.6.3.2 e 3.6.3.3. Cada um desses exemplos demonstra o processo de alinhamento de indivíduos, destacando como as correspondências entre valores específicos dos campos resultam em emparelhamentos entre os indivíduos.

#### 3.6.3.1 Exemplo de alinhamento de indivíduos

Neste exemplo, vamos dar continuidade ao processo de alinhamento de indivíduos, focando no alinhamento dos indivíduos presentes nos dois objetos destacados nos *Listings* 3.32 e 3.33, respectivamente. Nesse cenário específico, observamos uma correspondência entre os campos “Name” e “Names” nos dois objetos, e optaremos por realizar o alinhamento utilizando o método da distância de edição. A Figura 10 oferece uma representação visual das etapas do processo de alinhamento, bem como dos procedimentos inerentes a cada uma delas.

```
1 {
2     name: Pessoa1
3     parameters: [Names,Docs,birth]
4     entities: [
5         [John Smith,123456,1990-05-15],
```

```
6           [Emily Johnson,789012,1985-11-22],
7           [Michael Brown,345678,1978-03-10],
8           [Sophia Davis,901234,2001-09-03],
9           [Daniel Wilson,567890,1993-07-18]
10      ]
11      extension: csv
12      file_name: Pessoa1
13      match_name: []
14      match_parameters: [
15              {
16               name: Pessoa2,
17               matched_parameter: Name,
18               my_parameter: Names,
19               approximate': true
20          }
21      ]
22      match_entities: []
23  }
```

Listing 3.32: Objeto de base para Pessoa 1

```
1   {
2       name: Pessoa2
3       parameters: [Name,address,email]
4       entities: [
5           [John Smit,123 Main St,jhon@example.com],
6           [David Johnson,456 Elm Rd,david@example.com],
7           [Daniel Wilsons,789 Oak Ave,daniel@example.com],
8           [Christopher Lee,321 Pine Ln,christopher@example.com],
9           [Jennifer Davis,555 Maple Dr,jennifer@example.com]
10      ]
11      extension: csv
12      file_name: Pessoa2
13      match_name: []
14      match_parameters: []
15      match_entities: []
16  }
```

Listing 3.33: Objeto de base para Pessoa 2

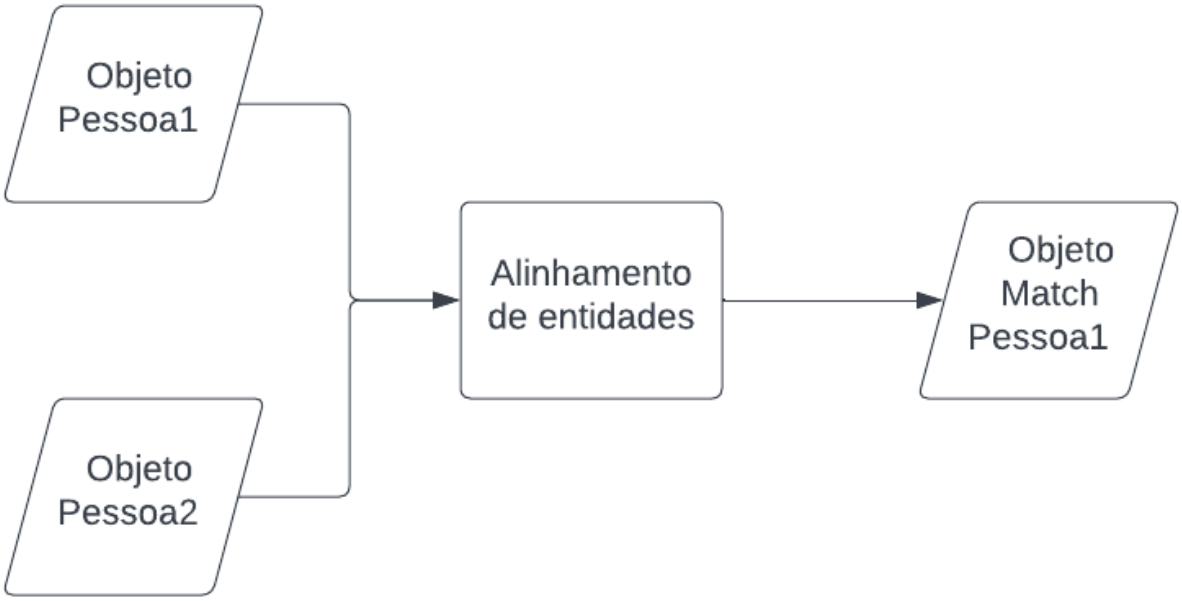


Figura 10: Fluxograma do alinhamento de pessoas.

No contexto específico em que o alinhamento dos campos se limitou aos atributos "Names" e "Name", a etapa subsequente de verificação de alinhamento entre os indivíduos restringirá sua análise somente aos valores presentes nessas colunas. Aqui, ocorrerá uma combinação desses valores por meio dos métodos de alinhamento selecionados pelo usuário e pelo método exato. Para cada correspondência identificada, os valores correspondentes serão marcados como tal no primeiro objeto de alinhamento. A abordagem começa pela avaliação dos elementos por meio do método exato, que neste caso resultará apenas em não correspondências. Na sequência, o algoritmo implementará o método de distância de edição. Consequentemente, serão identificados dois pares correspondentes: entre "John Smit" e "John Smith", bem como entre "Daniel Wilson" e "Daniel Wilsons". Esses resultados serão então registrados no *Listings* 3.32, culminando na produção do objeto apresentado no *Listings* 3.34.

```
1 {
2     name: Pessoa1
3     parameters: [Names,Docs,birth]
4     entities: [
5         [John Smith,123456,1990-05-15],
6         [Emily Johnson,789012,1985-11-22],
7         [Michael Brown,345678,1978-03-10],
8         [Sophia Davis,901234,2001-09-03],
```

```
9           [Daniel Wilson,567890,1993-07-18]
10      ]
11      extension: csv
12      file_name: Pessoa1
13      match_name: []
14      match_parameters: [
15          {
16              name: Pessoa2,
17              matched_parameter: Name,
18              my_parameter: Names,
19              approximate': true
20          }
21      ]
22      match_entities: [
23          {
24              matched_name: Pessoa2,
25              matched_entity_index: 0,
26              my_parameter_index: 0,
27          },
28          {
29              matched_name: Pessoa2,
30              matched_entity_index: 2,
31              my_parameter_index: 4,
32          }
33      ]
34  }
```

Listing 3.34: Objeto de base para Pessoa 1

#### 3.6.3.2 Exemplo de alinhamento de indivíduos

Neste exemplo, prosseguiremos com o alinhamento de DNA, direcionando nosso foco para o alinhamento dos indivíduos contidos nos dois objetos destacados nos *Listings* 3.35 e 3.36, respectivamente. Dentro deste cenário específico, observamos uma correspondência entre todos os campos presentes nos dois objetos, levando-nos a escolher a abordagem de alinhamento baseada em tradução. Para melhor visualização do processo de alinhamento e das etapas inerentes a ele, a Figura 11 proporciona uma representação gráfica deste processo.

```
{
    name: Dna1
    parameters: [sample_id,location,sample_type]
    entities: [
        [1,Laboratory A,Blood],
        [2,Hospital X,Urine],
        [3,Clinic Y,Saliva],
        [4,Research Facility Z,Tissue],
        [5,Pharmacy P,Blood]
    ]
    extension: csv
    file_name: Dna1
    match_name: []
    match_parameters: [
        {
            name: Dna2,
            matched_parameter: sample_id,
            my_parameter: sample_id,
            approximate': false
        },
        {
            name: Dna2,
            matched_parameter: localização,
            my_parameter: location,
            approximate': true
        },
        {
            name: Dna2,
            matched_parameter: sample_type,
            my_parameter: sample_type,
            approximate': false
        }
    ]
    match_entities: []
}
```

Listing 3.35: Objeto de base para Dna 1

```
{
    name: Dna2
    parameters: [sample_id,localização,sample_type]
    entities: [
        [1,Hospital A,Sangue],
        [2,Laboratório X,Urina],
        [3,Clínica Y,Saliva],
        [4,Instituto de Pesquisa Z,Tecido],
        [5,Farmácia P,Sangue]
    ]
    extension: csv
    file_name: Dna2
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.36: Objeto de base para Dna 2

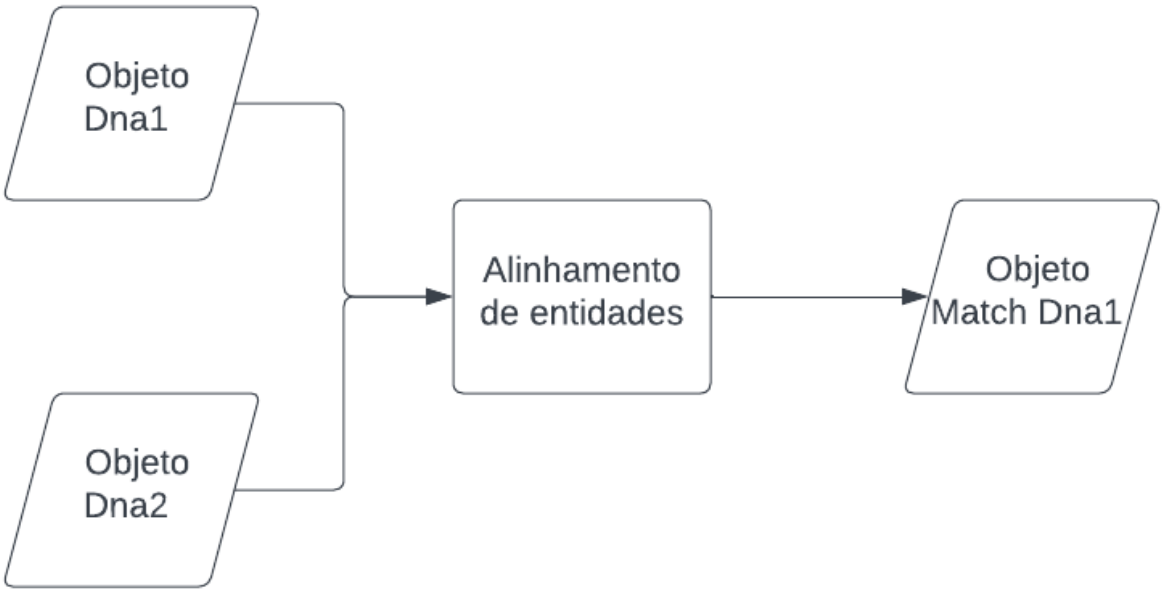


Figura 11: Fluxograma do alinhamento de dnas.

Dentro do contexto em que o alinhamento de campos identificou correspondências para todos os campos, a fase subsequente de verificação do alinhamento entre os indivíduos ex-

pandirá sua análise abrangendo todos os valores contidos em todos os campos. Nesse cenário, ocorrerá uma composição desses valores por meio dos métodos de alinhamento escolhidos pelo usuário, em conjunto com o método exato. Para cada correspondência estabelecida, os valores correspondentes serão anotados como tais no primeiro objeto de alinhamento. A abordagem se inicia pela avaliação dos elementos utilizando o método exato, que, nesse caso, resultará em valores verdadeiros para o campo "sample_id" e em valores negativos para os demais campos. Na sequência, o algoritmo empregará o método de tradução, gerando correspondências verdadeiras para alguns dos elementos dos campos "*location*" e "*localização*", bem como para os campos "*sample_type*" e "*sample_type*". Isso resultará na identificação de três pares correspondentes: referentes aos terceiro, quarto e quinto elementos nos dois objetos, respectivamente. Esses resultados serão então registrados no *Listings* 3.35, culminando na geração do objeto apresentado no *Listings* 3.37.

```
1  {
2      name: Dna1
3      parameters: [sample_id,location,sample_type]
4      entities: [
5          [1,Laboratory A,Blood],
6          [2,Hospital X,Urine],
7          [3,Clinic Y,Saliva],
8          [4,Research Facility Z,Tissue],
9          [5,Pharmacy P,Blood]
10     ]
11     extension: csv
12     file_name: Dna1
13     match_name: []
14     match_parameters: [
15         {
16             name: Dna2,
17             matched_parameter: sample_id,
18             my_parameter: sample_id,
19             approximate': false
20         },
21         {
22             name: Dna2,
23             matched_parameter: localização,
24             my_parameter: location,
25             approximate': true
```

```
26          },
27          {
28              name: Dna2,
29              matched_parameter: sample_type,
30              my_parameter: sample_type,
31              approximate': false
32          }
33      ]
34      match_entities: [
35          {
36              matched_name: Dna2,
37              matched_entity_index: 2,
38              my_parameter_index: 2,
39          },
40          {
41              matched_name: Dna2,
42              matched_entity_index: 3,
43              my_parameter_index: 3,
44          },
45          {
46              matched_name: Dna2,
47              matched_entity_index: 4,
48              my_parameter_index: 4,
49          }
50      ]
51  }
```

Listing 3.37: Objeto de base para Dna 1

#### 3.6.3.3 Exemplo de alinhamento de indivíduos

Neste exemplo, continuaremos com o processo de alinhamento de cidades, concentrando-nos no alinhamento dos indivíduos presentes nos objetos destacados nos *Listings* 3.38, 3.39 e 3.40, respectivamente. Dentro deste cenário específico, identificamos correspondências nos campos "Cidade" e "País" para os três objetos, tornando a abordagem de alinhamento restrita ao método exato. Para uma melhor compreensão do processo de alinhamento e das etapas nele envolvidas, a Figura 12 proporciona uma representação visual desse procedimento.

```
{
    name: Cidade1
    parameters: [Escritório,Cidade,País,Capacidade]
    entities: [
        [Escritório A,São Paulo,Brasil,100],
        [Escritório B,Paris,França,50],
        [Escritório C,Los Angeles,Estados Unidos,80],
        [Escritório D,Tóquio,Japão,70],
        [Escritório E,Londres,Reino Unido,60]
    ]
    extension: csv
    file_name: Cidade1
    match_name: []
    match_parameters: [
        {
            name: Cidade2,
            matched_parameter: Cidade,
            my_parameter: Cidade,
            approximate': false
        },
        {
            name: Cidade2,
            matched_parameter: País,
            my_parameter: País,
            approximate': false
        },
        {
            name: Cidade3,
            matched_parameter: Cidade,
            my_parameter: Cidade,
            approximate': false
        },
        {
            name: Cidade3,
            matched_parameter: País,
            my_parameter: País,
            approximate': false
```

```
38          }
39
40      ]
41      match_entities: []
42  }
```

Listing 3.38: Objeto de base para Cidade 1

```
1   {
2       name: Cidade2
3       parameters: [Filial,Cidade,País,Receita Anual]
4       entities: [
5           [Filial A,São Paulo,Brasil,5000000],
6           [Filial B,Nova York,Estados Unidos,8000000],
7           [Filial C,Paris,França,6000000],
8           [Filial D,Tóquio,Japão,4500000],
9           [Filial E,Londres,Reino Unido,7000000]
10      ]
11      extension: csv
12      file_name: Cidade2
13      match_name: []
14      match_parameters: [
15          {
16              name: Cidade3,
17              matched_parameter: Cidade,
18              my_parameter: Cidade,
19              approximate': false
20          },
21          {
22              name: Cidade3,
23              matched_parameter: País,
24              my_parameter: País,
25              approximate': false
26          }
27
28      ]
29      match_entities: []
30  }
```

Listing 3.39: Objeto de base para Cidade 2

```
{
    name: Cidade3
    parameters: [Ponto Turístico,Cidade,País,Visitantes Anuais]
    entities: [
        [Torre Eiffel,Paris,França,7000000],
        [Estátua da Liberdade,Nova York,Estados Unidos,5000000],
        [Machu Picchu,Cusco,Peru,1200000],
        [Pirâmides de Gizé,Cairo,Egito,900000],
        [Coliseu de Roma,Roma,Itália,6000000]
    ]
    extension: csv
    file_name: Cidade3
    match_name: []
    match_parameters: []
    match_entities: []
}
```

Listing 3.40: Objeto de base para Cidade 3

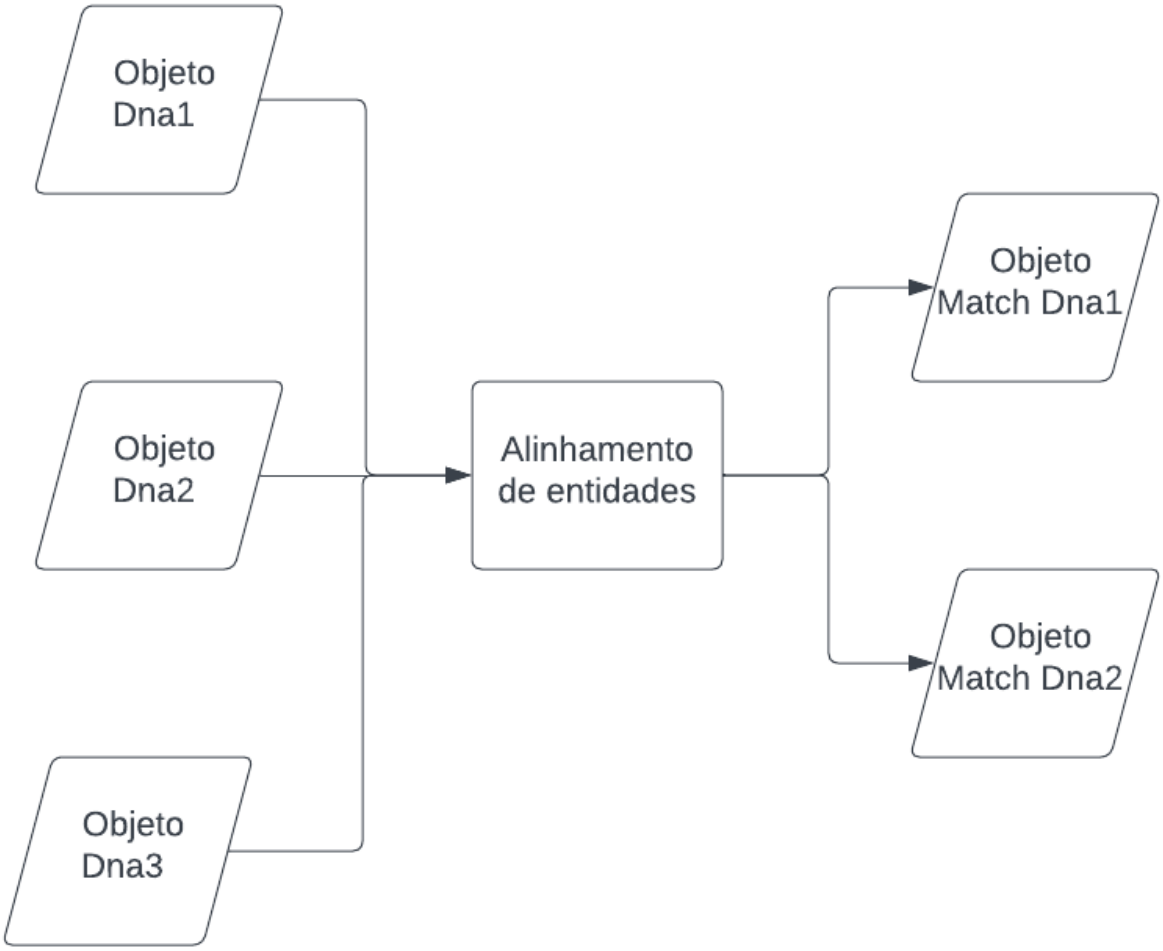


Figura 12: Fluxograma do alinhamento de cidades.

No cenário em que o alinhamento dos campos identificou correspondências nos campos "Cidade" e "País", a fase subsequente de verificação do alinhamento entre os indivíduos abarcará a análise dos valores contidos nesses campos. Nesse contexto, os valores serão alinhados por meio da aplicação do método exato. A cada correspondência estabelecida, os valores correspondentes serão anotados como tal no primeiro objeto de alinhamento. A abordagem se dá por meio da avaliação dos elementos, empregando o método exato, o qual, nesse caso, resultará em avaliações verdadeiras para ambos os campos. Esse procedimento culminará na identificação de quatro pares correspondentes entre os objetos de "Cidade1" e "Cidade2", um par correspondente entre os objetos de "Cidade1" e "Cidade3", bem como dois pares correspondentes entre os objetos de "Cidade2" e "Cidade3". Esses resultados serão então documentados nos *Listings* 3.38 e 3.39, resultando na geração dos objetos apresentados nos *Listings* 3.41 e 3.42.

```
{
    name: Cidade1
    parameters: [Escritório,Cidade,País,Capacidade]
    entities: [
        [Escritório A,São Paulo,Brasil,100],
        [Escritório B,Paris,França,50],
        [Escritório C,Los Angeles,Estados Unidos,80],
        [Escritório D,Tóquio,Japão,70],
        [Escritório E,Londres,Reino Unido,60]
    ]
    extension: csv
    file_name: Cidade1
    match_name: []
    match_parameters: [
        {
            name: Cidade2,
            matched_parameter: Cidade,
            my_parameter: Cidade,
            approximate': false
        },
        {
            name: Cidade2,
            matched_parameter: País,
            my_parameter: País,
            approximate': false
        },
        {
            name: Cidade3,
            matched_parameter: Cidade,
            my_parameter: Cidade,
            approximate': false
        },
        {
            name: Cidade3,
            matched_parameter: País,
            my_parameter: País,
            approximate': false
        }
```

```
39
40      ]
41      match_entities: [
42              {
43              matched_name: Cidade2,
44              matched_entity_index: 0,
45              my_parameter_index: 0,
46          },
47          {
48              matched_name: Cidade2,
49              matched_entity_index: 2,
50              my_parameter_index: 1,
51          },
52          {
53              matched_name: Cidade2,
54              matched_entity_index: 3,
55              my_parameter_index: 3,
56          },
57          {
58              matched_name: Cidade2,
59              matched_entity_index: 4,
60              my_parameter_index: 4,
61          },
62          {
63              matched_name: Cidade3,
64              matched_entity_index: 0,
65              my_parameter_index: 1,
66          }
67      ]
68  }
```

Listing 3.41: Objeto de base para Cidade 1

```
1  {
2      name: Cidade2
3      parameters: [Filial,Cidade,País,Receita Anual]
4      entities:  [
5          [Filial A,São Paulo,Brasil,5000000],
6          [Filial B,Nova York,Estados Unidos,8000000],
```

```
7           [Filial C,Paris,França,6000000],
8           [Filial D,Tóquio,Japão,4500000],
9           [Filial E,Londres,Reino Unido,7000000]
10      ]
11      extension: csv
12      file_name: Cidade2
13      match_name: []
14      match_parameters: [
15          {
16              name: Cidade3,
17              matched_parameter: Cidade,
18              my_parameter: Cidade,
19              approximate': false
20          },
21          {
22              name: Cidade3,
23              matched_parameter: País,
24              my_parameter: País,
25              approximate': false
26          }
27
28      ]
29      match_entities: [
30          {
31              matched_name: Cidade3,
32              matched_entity_index: 1,
33              my_parameter_index: 1,
34          },
35          {
36              matched_name: Cidade3,
37              matched_entity_index: 0,
38              my_parameter_index: 2,
39          }
40      ]
41  }
```

Listing 3.42: Objeto de base para Cidade 2

### 3.6.4 Geração de Resultados

A fase final do algoritmo é iniciada com a formação de um arquivo *CSV* alinhado, intitulado "bigbase", que engloba todos os campos e indivíduos presentes nas tabelas fornecidas pelo usuário. A construção deste arquivo tem início com a enumeração de todos os campos em todas as tabelas, posicionando-os na primeira linha do arquivo. Além disso, é adicionado o nome da tabela como prefixo a cada campo. Por exemplo, considerando uma tabela chamada "tabela1" com o campo "campo1", este será registrado como "tabela1_campo1". Essa lista é criada sequencialmente, começando com todos os campos da tabela 1, seguida pelos da tabela 2 e assim por diante até a tabela "n", sendo todos os elementos separados por vírgulas. Posteriormente, para cada indivíduo detectado nas tabelas, eles são incorporadas ao arquivo *CSV*, com cada indivíduo ocupando uma linha individual.

O processo de posicionamento dos indivíduos ocorre através da verificação de seus campos, dispostos de forma a associar cada propriedade do indivíduo à sua respectiva posição no campo correspondente. Cada correspondência identificada resulta na inclusão do indivíduo alinhado nos campos pertinentes de sua tabela. No entanto, caso algum campo não apresente correspondência ou tenha um valor ausente, esse campo será preenchido com o valor "null". Para ilustrar esse processo, as Tabelas 1, 2 e 3 prosseguem com o processo de alinhamento já observado nos exemplos anteriores de "Pessoa", "Dna" e "Cidade", exibindo os resultados de seus respectivos "bigbases" após o alinhamento.

| **Pessoa1_Names** | **Pessoa1_Docs** | **Pessoa1_birth** | **Pessoa2_Name** | **Pessoa2_address** | **Pessoa2_email** |
|---|---|---|---|---|---|
| John Smith | 123456 | 1990-05-15 | John Smit | 123 Main St | jhon@example.com |
| Emily Johnson | 789012 | 1985-11-22 | null | null | null |
| Michael Brown | 345678 | 1978-03-10 | null | null | null |
| Sophia Davis | 901234 | 2001-09-03 | null | null | null |
| Daniel Wilson | 567890 | 1993-07-18 | Daniel Wilsons | 789 Oak Ave | daniel@example.com |
| null | null | null | David Johnson | 456 Elm Rd | david@example.com |
| null | null | null | Christopher Lee | 321 Pine Ln | christopher@example.com |
| null | null | null | Jennifer Davis | 555 Maple Dr | jennifer@example.com |

Tabela 1: Bigbase Pessoa

| **Dna1_ sample_id** | **Dna1_ location** | **Dna1_sample _type** | **Dna2_ sample_id** | **Dna2_ localização** | **Dna2_sample _type** |
|---|---|---|---|---|---|
| 1 | Laboratory A | Blood | null | null | null |
| 2 | Hospital X | Urine | null | null | null |
| 3 | Clinic Y | Saliva | 3 | Clínica Y | Saliva |
| 4 | Research Facility Z | Tissue | 4 | Instituto de Pesquisa Z | Tecido |
| 5 | Pharmacy P | Blood | 5 | Farmácia P | Sangue |
| null | null | null | 1 | Hospital A | Sangue |
| null | null | null | 2 | Laboratório X | Urina |

Tabela 2: Bigbase Dna

| **Cidade1_ Escritório** | **Cidade1_ Cidade** | **Cidade1_País** | **Cidade1_ Capacidade** | **Cidade2_ Filial** | **Cidade2_ Cidade** | **Cidade2_País** | **Cidade2_l Receita Anual** | **Cidade3_ Ponto Turístico** | **Cidade3_ Cidade** | **Cidade3_País** | **Cidade3_ Visitantes Anuais** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Escritório A | São Paulo | Brasil | 100 | Filial A | São Paulo | Brasil | 5000000 | null | null | null | null |
| Escritório B | Paris | França | 50 | Filial C | Paris | França | 6000000 | Torre Eiffel | Paris | França | 7000000 |
| Escritório C | Los Angeles | Estados Unidos | 80 | null | null | null | null | null | null | null | null |
| Escritório D | Tóquio | Japão | 70 | Filial D | Tóquio | Japão | 4500000 | null | null | null | null |
| Escritório E | Londres | Reino Unido | 60 | Filial E | Londres | Reino Unido | 7000000 | null | null | null | null |
| null | null | null | null | Filial B | Nova York | Estados Unidos | 8000000 | Estátua da Liberdade | Nova York | Estados Unidos | 5000000 |
| null | null | null | null | null | null | null | null | Machu Picchu | Cusco | Peru | 1200000 |
| null | null | null | null | null | null | null | null | Pirâmides de Gizé | Cairo | Egito | 900000 |
| null | null | null | null | null | null | null | null | Coliseu de Roma | Roma | Itália | 6000000 |

Tabela 3: Bigbase Cidade

Após a criação do arquivo alinhado, o algoritmo desdobra-se em três processos distintos: a integração com a SUMO, a consulta com o Pandas e, por fim, o aprimoramento por meio do Deep Matcher. No estágio de integração com a SUMO, o algoritmo inicia traduzindo todos os campos das tabelas para o inglês, garantindo assim a correspondência de idiomas com a Wordnet. Em seguida, para cada campo traduzido, é realizada uma verificação utilizando a distância de edição, na tentativa de encontrar elementos correspondentes na Wordnet. Consequentemente, um objeto é construído para conter os resultados desses dados. No entanto, caso não se encontre nenhum elemento correspondente na Wordnet para um dado campo, o objeto é preenchido com valores "null", indicando a ausência de dados. O objeto mencionado pode ser visualizado no *Listing* 3.43 e exemplos de objeto gerado para "Pessoa", "Dna" e "Cidade" estão descritos nos *Listings* 3.44, 3.45 e 3.46.

```
1    word: "Representa a palavra original"
2    translation: "Representa a tradução da palavra"
3    code: "Representa o código da palavra encontrada na Wordnet"
4    syn: "Representa a palavra encontrada na Wordnet"
5    relation: "Representa o tipo de relação da palavra"
6    meaning: "O significado da palavra dito pela Wordnet"
```

Listing 3.43: Objeto da SUMO

```
1    word: Names
2    translation: Names
3    code: 106720784
4    syn: Expressing
5    relation: subsumed
6    meaning: Verbal abuse; a crude substitute for argument; "sticks
        and stones may break my bones but names can never hurt me"
```

Listing 3.44: Objeto da SUMO Pessoa

```
1    word: location
2    translation: location
3    code: 100155487
4    syn: Learning
5    relation: subsumed
6    meaning: A determination of the place where something is; "he got
         a good fix on the target"
```

Listing 3.45: Objeto da SUMO Dna

```
1    word: Escritório
2    translation: Desk
3    code: 103179701
4    syn: Desk
5    relation: equivalent
6    meaning: A piece of furniture with a writing surface and usually
        drawers or other compartments
```

Listing 3.46: Objeto da SUMO Cidade

Posteriormente, um arquivo *CSV* que reúne todos os dados obtidos nessa fase é criado, denominado "sumo_alignment". Sua estrutura é organizada da seguinte maneira: a primeira linha contém os campos extraídos do objeto da SUMO, representados por *word*, *translation*, *code*, *synonym*, *relation* e *meaning*. Para cada objeto gerado, os valores correspondentes são preenchidos nos campos equivalentes, resultando em uma linha no arquivo. A fim de ilustrar, três exemplos que exemplificam os dados referentes a "Pessoa", "Dna" e "Cidade" estão apresentados nas Tabelas 4, 5 e 6.

| **word** | **translation** | **code** | **synonym** | **relation** | **meaning** |
|---|---|---|---|---|---|
| Names | Names | 106720784 | Expressing | subsumed | Verbal abuse; a crude substitute for argument; "sticks and stones may break my bones but names can never hurt me" |
| Docs | Docs | null | null | null | null |
| birth | birth | 107320302 | Birth | equivalent | The event of being born; "they celebrated the birth of their first child" |
| Name | Name | 106333653 | Name | equivalent | A language unit by which a person or thing is known; "his name really is George Washington"; "those are two names for the same thing" |
| address | address | 201033527 | Process | subsumed | Act on verbally or in some form of artistic expression; "This book deals with incest"; "The course covered all of Western Civilization"; "The new book treats the history of China" |
| email | email | 201032451 | Emailing | equivalent | Communicate electronically on the computer; "she e-mailed me the good news" |

Tabela 4: Objeto da SUMO Pessoa

| word | translation | code | synonym | relation | meaning |
|---|---|---|---|---|---|
| sample_id | sample_id | null | null | null | null |
| location | location | 100155487 | Learning | subsumed | A determination of the place where something is; "he got a good fix on the target" |
| sample_type | sample_type | null | null | null | null |
| localização | location | 100155487 | Learning | subsumed | A determination of the place where something is; "he got a good fix on the target" |
| sample_type | sample_type | null | null | null | null |

Tabela 5: Objeto da SUMO Dna

| word | translation | code | synonym | relation | meaning |
|---|---|---|---|---|---|
| Escritório | Desk | 103179701 | Desk | equivalent | A piece of furniture with a writing surface and usually drawers or other compartments |
| Cidade | City | 108226335 | City | subsumed | People living in a large densely populated municipality; "the city voted for Republicans in 1994" |
| País | Country | 108166552 | Group | subsumed | The people who live in a nation or country; "a statement that sums up the nation's mood"; "the news was announced to the nation"; "the whole country worshipped him" |
| Capacidade | Capacity | 100720951 | Position | subsumed | A specified function; "he was employed in the capacity of director"; "he should be retained in his present capacity at a higher salary" |
| Filial | Branch | 113163250 | PlantBranch | equivalent | A division of a stem, or secondary stem arising from the main stem of a plant |
| Receita Anual | Annual recipe | null | null | null | null |
| Ponto Turístico | Tourist spot | null | null | null | null |
| Visitantes Anuais | Annual Visitors | null | null | null | null |

Tabela 6: Objeto da SUMO Cidade

A conexão com a SUMO é estabelecida por meio da ferramenta SIGMA-KEE. Nesse processo, cada campo identificado na WordNet é associado aos valores correspondentes

dos indivíduos, que são incorporados como instâncias desses campos na ontologia. Entretanto, caso alguns campos não sejam encontrados como correspondência na SUMO, eles são inseridos como novos campos, e seus indivíduos são conectados a esses novos campos. Adicionalmente, novas funções são introduzidas para representar o conceito destes indivíduos e suas posições na tabela. Esse método garante que os usuários possam conduzir inferências em relação aos seus dados na SUMO.

Ao final desse estágio, o algoritmo executa as inferências seguindo a ordem alfabética das encontradas na pasta "*inferences*". Cada inferência realizada resulta na geração de um arquivo de inferência na pasta "*results*" com o nome "*inferences_index*", em que o índice indica a posição em que a inferência foi identificada.

Os exemplos de Pessoa, DNA e Cidade ilustrados nos *Listings* 4, 5 e 6 demonstram inferências que podem ser aplicadas. Esse processo amplia a capacidade de dedução sobre os dados, possibilitando uma compreensão mais abrangente no contexto da SUMO.

```
1 Return a person with the email jhon@example.com
```

Listing 3.47: Inferência Pessoa

```
1 List all the sample_type with value Blood
```

Listing 3.48: Inferência Dna

```
1 List all the cities with visitantes anuais greather than 500000
```

Listing 3.49: Inferência Cidade

Após a conclusão do processo de integração com o SUMO, os procedimentos de consulta utilizando o Pandas são executados. Nesse contexto, o algoritmo examina a pasta denominada "*queries*" em busca de consultas válidas nos formatos booleano ou SQL. Posteriormente, as consultas são executadas em ordem alfabética. Para cada consulta válida identificada, um arquivo é gerado na pasta "*results*", com o nome "*queries_index*", em que o índice representa a posição em que a consulta foi efetuada, começando a contagem a partir do valor zero. A lista de comparações booleanas está disponível a seguir.

- $<$ (Menor que)
- $>$ (Maior que)
- $<=$ (Menor igual que)

- >= (Maior igual que)
- == (Igualdade)
- ! = (Diferença)
- & ("E" lógico)
- | ("Ou" lógico)
- ∼ (Negação lógica)

Cada consulta necessita ser realizada mediante a inserção do nome do arquivo como um campo, sendo que a base para essas consultas é o arquivo denominado bigbase. A aplicação prática deste processo pode ser visualizada nos exemplos de consultas sobre os indivíduos como Pessoa, DNA e Cidade, os quais são ilustrados nos *Listings* 3.50, 3.51 e 3.52.

```
Pessoa1_Names == "John Smith"
```

Listing 3.50: Consulta Pessoa

```
SELECT Dna1_sample_id,
       Dna1_location,
       Dna1_sample_type,
       Dna2_sample_id,
       Dna2_location,
       Dna2_sample_type
WHERE Dna1_type = 'Blood' AND Dna2_sample_type = 'Sangue'
ORDER BY Dna1_sample_id;
```

Listing 3.51: Consulta Dna

```
SELECT
    Cidade1_Escritório,
    Cidade1_Cidade,
    Cidade1_País,
    Cidade1_Capacidade,
    Cidade2_Filial,
    Cidade2_Cidade,
    Cidade2_País,
    Cidade2_Receita_Anual,
```

```
10        Cidade3_Ponto_Turístico ,
11        Cidade3_Cidade ,
12        Cidade3_País ,
13        Cidade3_Visitantes_Anuais
14 WHERE Cidade2_Receita_Anual > 500000
15 ORDER BY Cidade1_.Escritório;
```

Listing 3.52: Consulta Cidade

Os resultados destas consultas estão disponíveis nos *Listings* 3.50, 3.51 e 3.52. Além disso, após a conclusão da execução do programa, é possível realizar novas inferências sobre o arquivo. Para isso, o usuário deve executar o comando "python -i loading", o que abrirá o console Python. A partir daí, as consultas listadas na Lista 3.6.4 podem ser realizadas, sendo necessário executar o comando apropriado para exportar cada consulta como um arquivo *CSV* após o término de cada consulta.

| **Pessoa1_Names** | **Pessoa1_Docs** | **Pessoa1_birth** | **Pessoa2_Name** | **Pessoa2_address** | **Pessoa2_email** |
|---|---|---|---|---|---|
| John Smith | 123456 | 1990-05-15 | John Smit | 123 Main St | jhon@example.com |

Tabela 7: Consulta Pessoa

| **Dna1_ sample_id** | **Dna1_ location** | **Dna1_sample _type** | **Dna2_ sample_id** | **Dna2_ localização** | **Dna2_sample _type** |
|---|---|---|---|---|---|
| 5 | Pharmacy P | Blood | 5 | Farmácia P | Sangue |

Tabela 8: Consulta Dna

| **Cidade1_ Escritório** | **Cidade1_ Cidade** | **Cidade1_País** | **Cidade1_ Capacidade** | **Cidade2_ Filial** | **Cidade2_ Cidade** | **Cidade2_País** | **Cidade2_l Receita Anual** | **Cidade3_ Ponto Turístico** | **Cidade3_ Cidade** | **Cidade3_País** | **Cidade3_ Visitantes Anuais** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Escritório B | Paris | França | 50 | Filial C | Paris | França | 6000000 | Torre Eiffel | Paris | França | 7000000 |
| Escritório E | Londres | Reino Unido | 60 | Filial E | Londres | Reino Unido | 7000000 | null | null | null | null |
| null | null | null | null | Filial B | Nova York | Estados Unidos | 8000000 | Estátua da Liberdade | Nova York | Estados Unidos | 5000000 |

Tabela 9: Consulta Cidade

- pandaSQL("consulta") - Realiza uma consulta no formato SQL.
- pandaBoolean("consulta") - Realiza uma consulta no formato booleano.
- export2CSV("arquivo.csv") - Salva as consultas realizadas como um arquivo *CSV*.

O aprimoramento por meio do Deep Matcher é uma abordagem alternativa que é ativada somente quando seu comando é executado. Para isso, o processo de alinhamento precisa ter sido previamente concluído. Caso os dados não tenham sido adequadamente alinhados, essa etapa pode ser acionada para refinamento por meio do algoritmo de Deep Matcher. Para executar essa ação, é necessário utilizar o comando "python deep.py".

# 4 Interface

A implementação desta interface foi desenvolvida utilizando Next.js em conjunto com o Node 18.12.1, npm 9.6.7 e yarn 1.22.18. O código-fonte do projeto está disponível no seguinte repositório do GitHub: https://github.com/frame-lab/interoperaNit.

Para executar a versão atual da interface, é necessário que o processo de instalação da ferramenta tenha sido concluído e que ela esteja pronta para uso. Além disso, as versões dos pacotes instalados deve ser equivalente as listadas na Lista 4. Com essas etapas cumpridas, a interface estará pronta para ser utilizada.

- next: latest
- prop-types: ^15.8.1
- react: 17.0.2
- react-dom: 17.0.2
- react-dropzone: ^12.0.4
- styled-components: ^5.3.5
- @babel/core: ^7.18.6
- @babel/eslint-parser: ^7.17.0
- eslint: ^8.12.0
- eslint-config-airbnb: ^19.0.4
- eslint-config-next: ^12.1.4
- eslint-config-prettier: ^8.5.0
- eslint-plugin-import: ^2.25.4

- eslint-plugin-jsx-a11y: ˆ6.5.1
- eslint-plugin-prettier: ˆ4.0.0
- eslint-plugin-react: ˆ7.29.4
- eslint-plugin-react-hooks: ˆ4.4.0
- prettier: ˆ2.6.2

Para instalar os pacotes necessários, pode-se optar por utilizar os seguintes comandos individuais para cada pacote: "npm install nome_do_pacote" ou "yarn add nome_do_pacote". Alternativamente, pode-se instalar todos os pacotes de uma só vez, executando o comando "npm install" ou "yarn add". É importante destacar que é fundamental escolher apenas um tipo de comando para instalar todos os pacotes necessários, garantindo assim uma instalação coesa e eficiente das dependências para o correto funcionamento da ferramenta.

Após a instalação dos pacotes necessários, para executar a interface, devem ser seguidos os passos abaixo, de acordo com o instalador escolhido anteriormente: Utilize o comando "npm run build" ou "yarn build" para construir a interface. Em seguida, utilize o comando "npm run start" ou "yarn start" para iniciar a interface. Esses comandos iniciarão a interface e permitirão o uso da ferramenta de forma adequada. Certifique-se de seguir essa sequência para garantir o correto funcionamento da interface. Com isso, todas as funcionalidades da ferramenta estarão prontas para serem exploradas de maneira eficiente e sem problemas.

Ao executar a ferramenta, caso nenhum parâmetro extra seja especificado, a interface será aberta na porta 3000 do sistema operacional, podendo ser acessada através do link `http://localhost:3000`. No entanto, se a porta estiver ocupada, um erro será exibido no terminal em que o comando foi executado. Uma alternativa é utilizar o parâmetro "–port numero_da_porta" ao final do comando para executar a interface em uma porta diferente. Nesse caso, a URL de acesso seria `http://localhost:numero_da_porta`, correspondendo à porta especificada no parâmetro.

Ao acessar a interface, o usuário será direcionado à tela inicial, denominada *home*, como ilustrado na Figura 13. Nessa tela, serão apresentadas diversas opções e campos a serem preenchidos, caso o usuário opte por passar pelo processo completo de alinhamento padrão do algoritmo. No entanto, caso deseje executar apenas as etapas de pós-processamento, basta selecionar a opção "*POST PROCESSING*".

O processo de configuração dos arquivos e opções pela interface do processo padrão é dividido em 5 etapas. No primeiro passo, na tela inicial, todas as opções para realizar o alinhamento das bases de dados estarão disponíveis. Cada uma dessas opções executará o algoritmo de alinhamento, detalhado no Capítulo 3, através de um comando equivalente. Por exemplo, a opção *Synonym* corresponderá ao comando "-s" do algoritmo, enquanto a opção *Translate* estará associada ao comando "-t".

Figura 13: Página de Home da interface

Esta tela também inclui as opções *unique*, *split* e *approximate*. Essas opções têm a finalidade de permitir o preenchimento dos arquivos de forma direta pela interface. Ao preencher cada entrada, uma linha correspondente será adicionada ao arquivo correspondente. Caso seja necessário escrever múltiplas linhas para um arquivo específico, basta adicionar novas entradas clicando no botão de "(+)". Dessa maneira, o usuário pode facilmente inserir e organizar o conteúdo nos arquivos conforme necessário.

Caso tenha sido adicionado um novo serviço à ferramenta, é necessário incluir uma nova opção na interface para representar esse serviço. Para fazer isso, adicione a nova opção à variável "options" no arquivo "index" localizado na pasta "alignment". O caminho relativo para encontrar essa pasta é "components\features\alignment". Isso garantirá que o novo serviço seja devidamente refletido e acessível por meio da interface da ferramenta.

Um simples exemplo do primeiro passo de alinhamento preenchido pode ser vista na Figura 14.

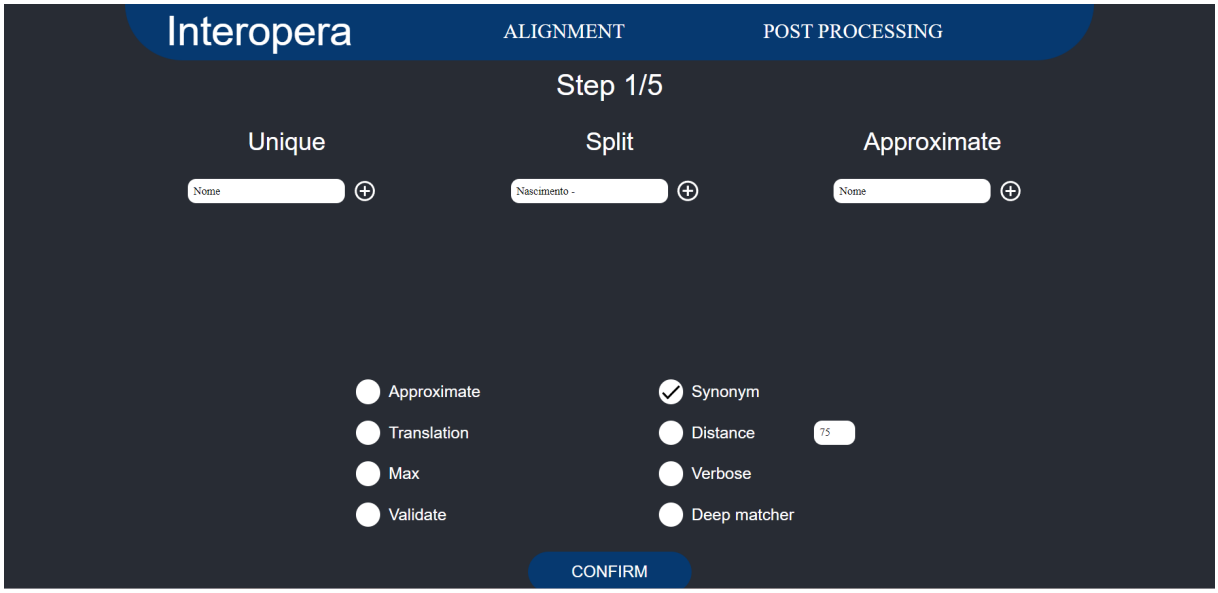


Figura 14: Página de Home da interface preenchida

Após escolher todas as opções desejadas e preencher as entradas necessárias, para dar continuidade ao processo de execução, basta clicar no botão "*CONFIRM*". Esse botão conduzirá o usuário à página de seleção de arquivos, como exemplificado na Figura 15. Nesta página, é necessário selecionar todos os arquivos que serão utilizados no processo de alinhamento, clicando no botão para adicionar os arquivos. Os arquivos selecionados serão exibidos em uma lista na interface, e é possível remover arquivos clicando no botão de "(-)" ao lado de cada item da lista. Além disso, é permitido adicionar mais arquivos através do botão de adição. Ao finalizar a seleção, a tela deverá apresentar uma configuração similar à ilustrada na Figura 16.

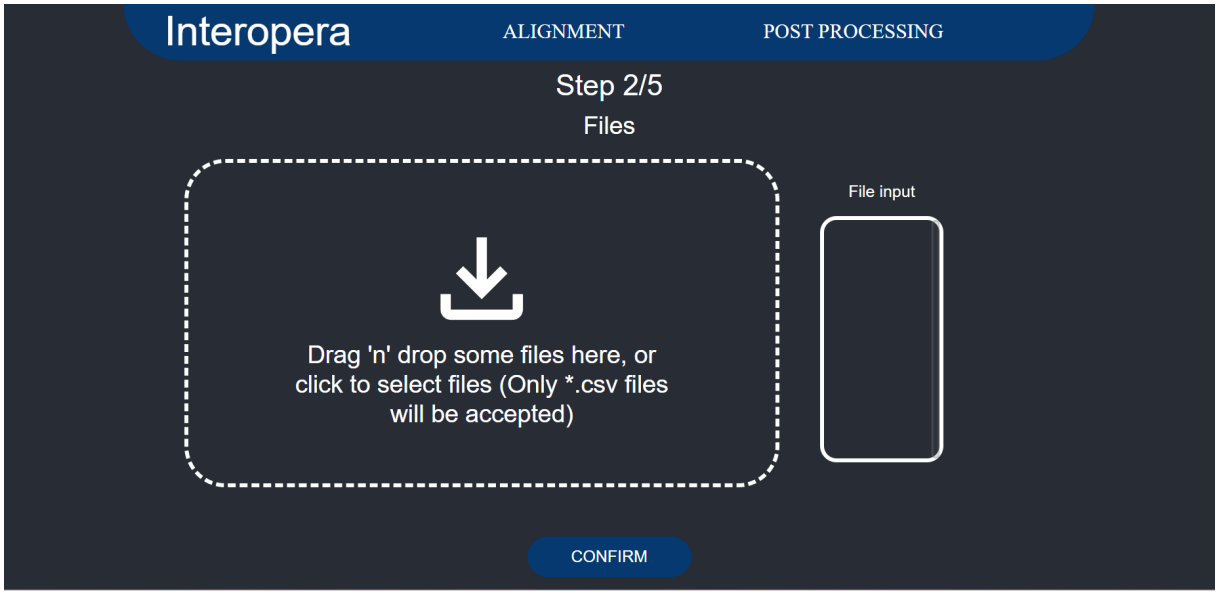


Figura 15: Página de seleção de arquivos da interface

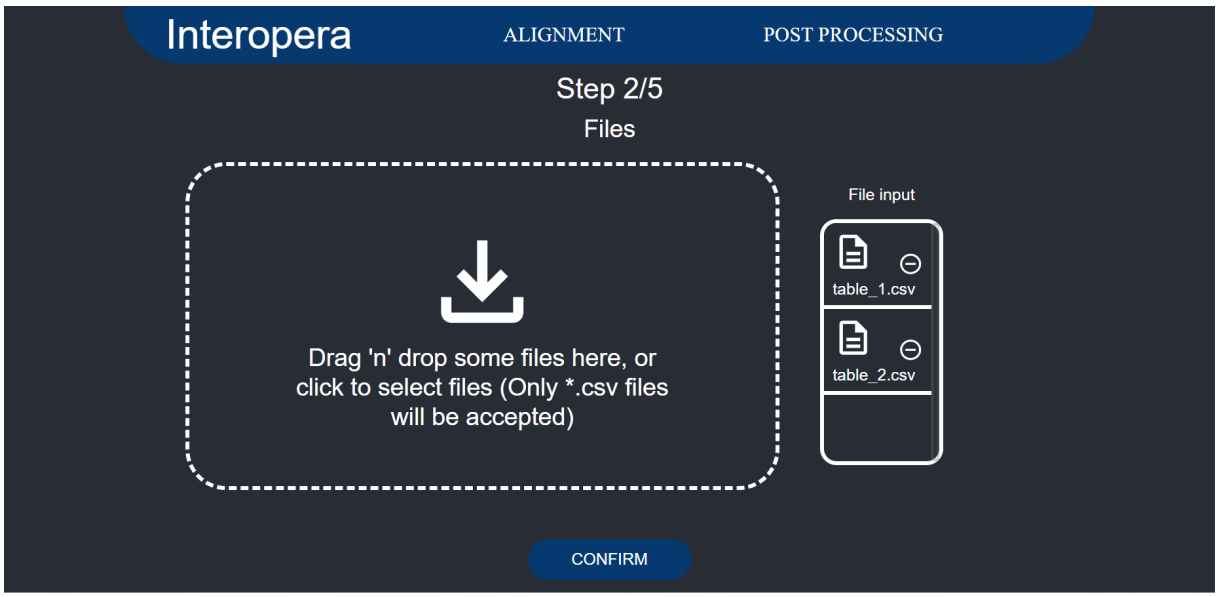


Figura 16: Página de seleção de arquivos da interface preenchida

Essa configuração resultará no preenchimento da pasta "Samples", conforme mencionada no Capítulo 3, com os arquivos selecionados na interface. É importante destacar que, caso a pasta "samples" já contenha arquivos previamente armazenados, esses arquivos serão apagados, e apenas os novos arquivos selecionados na interface serão adicionados à pasta.

Com os arquivos selecionados, a continuação da execução da interface continua ao cli-

car no botão "*CONFIRM*", que direciona o usuário para a página de seleção de consultas, conforme ilustrado na Figura 17. Nesta página, é possível configurar todas as consultas que o usuário deseja executar ao final do processo de alinhamento dos arquivos. Essas consultas são as mesmas mencionadas no Capítulo 3 e podem ser realizadas em qualquer um dos dois formatos previamente citados: *SQL* e *Booleano*. Importante destacar que a execução dessas consultas é opcional para prosseguir com a execução. Um exemplo simples das consultas que podem ser realizadas é apresentado na Figura 18.

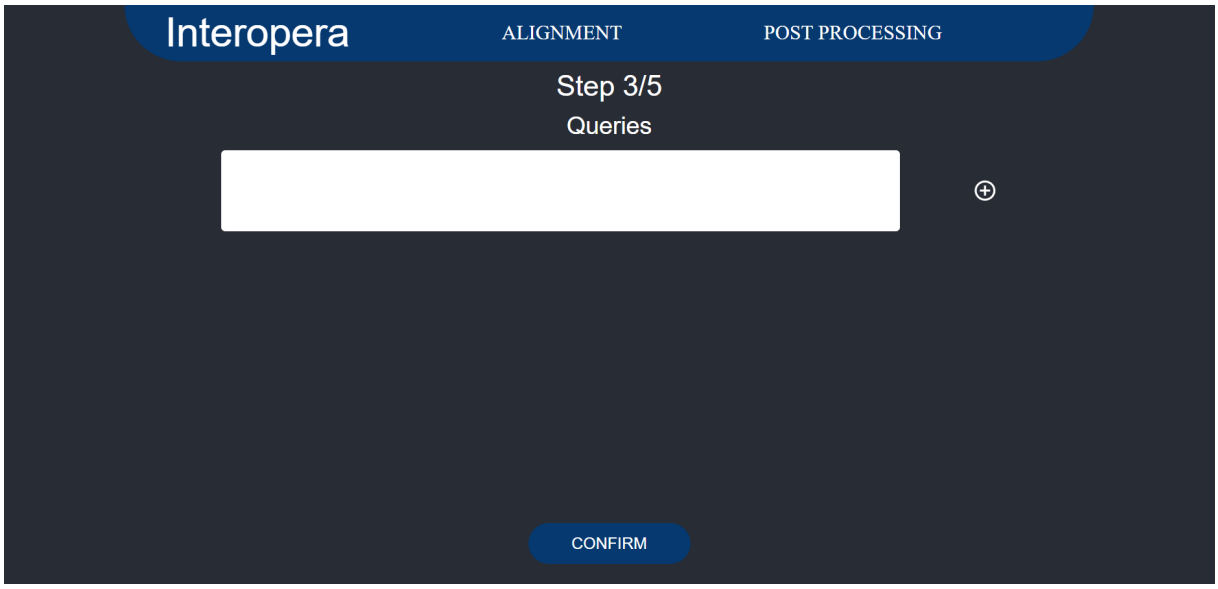


Figura 17: Página de seleção de consultas da interface

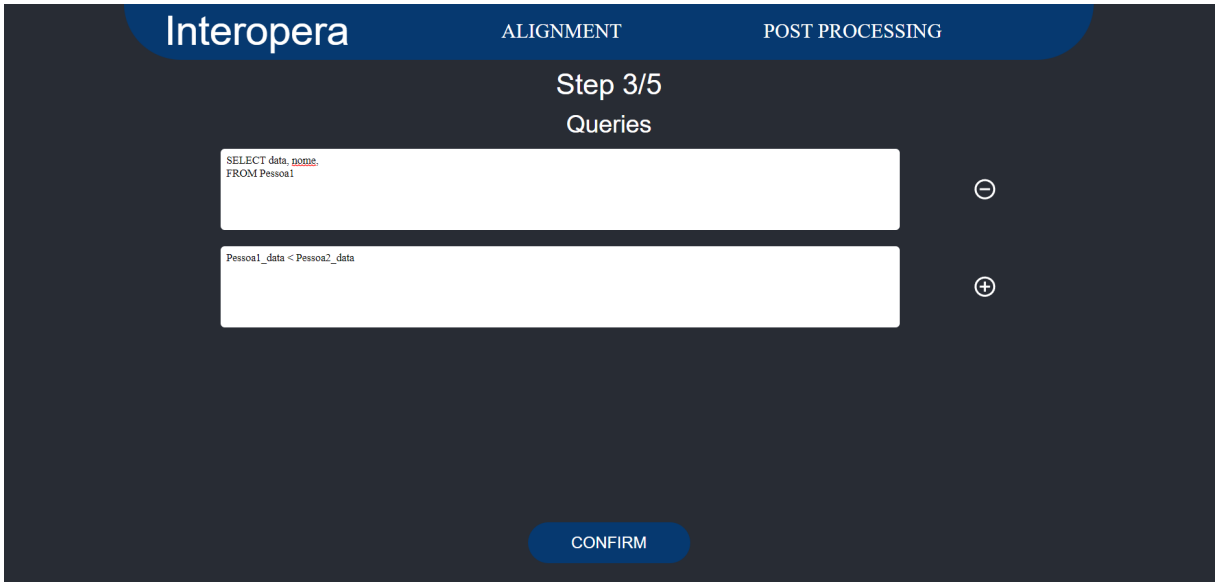


Figura 18: Página de seleção de consultas da interface preenchida

Após concluir a execução da interface, conforme exemplificado na Figura 18, a interface irá criar dois arquivos de consulta na pasta "queries", e o algoritmo produzirá duas consultas resultantes ao fim de seu processamento: uma no formato *SQL*, armazenada no arquivo denominado *query_1*, e outra em formato booleano, salva como *query_2* na pasta "*results*", conforme mencionado no Capítulo 3. Para finalizar a seleção de consultas, basta clicar no botão "*CONFIRM*", o que redirecionará a interface para a tela de seleção de inferências, como ilustrado na Figura 19.

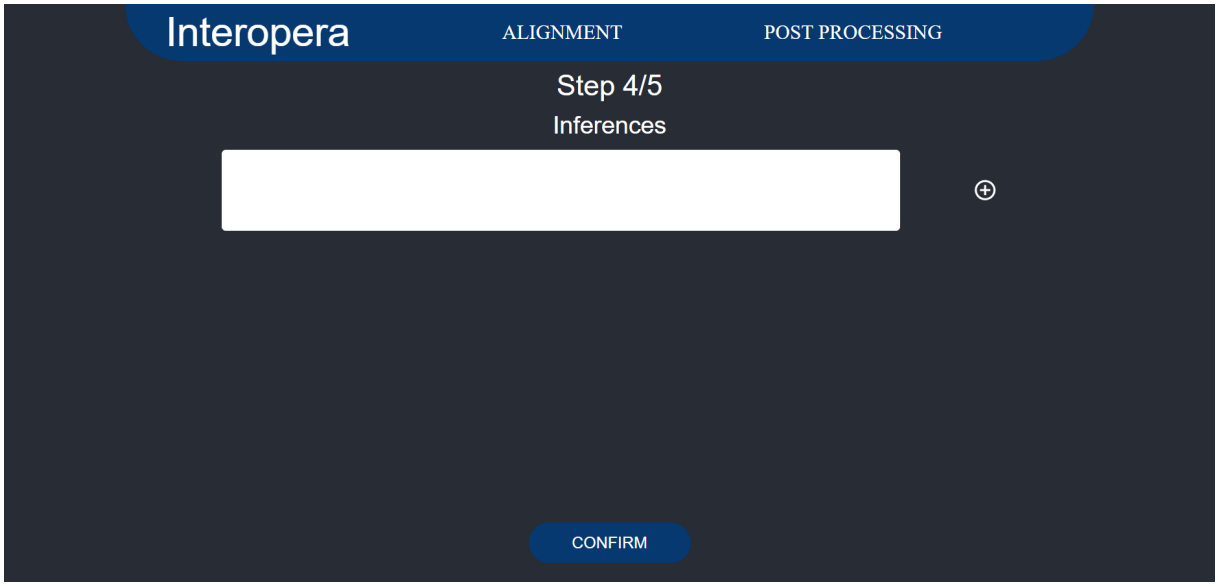


Figura 19: Página de seleção de inferências da interface

Nesta página, é possível configurar todas as inferências que devem ser executadas ao final do processo de alinhamento dos arquivos. Um exemplo simples das consultas que podem ser realizadas é apresentado na Figura 20. Nesse exemplo, temos duas inferências sobre pessoas que serão criadas na pasta "inferences". Ao final do processamento, serão gerados dois arquivos resultantes: "inferences_1" e "inferences_2", na pasta "*results*", conforme mencionado no Capítulo 3.

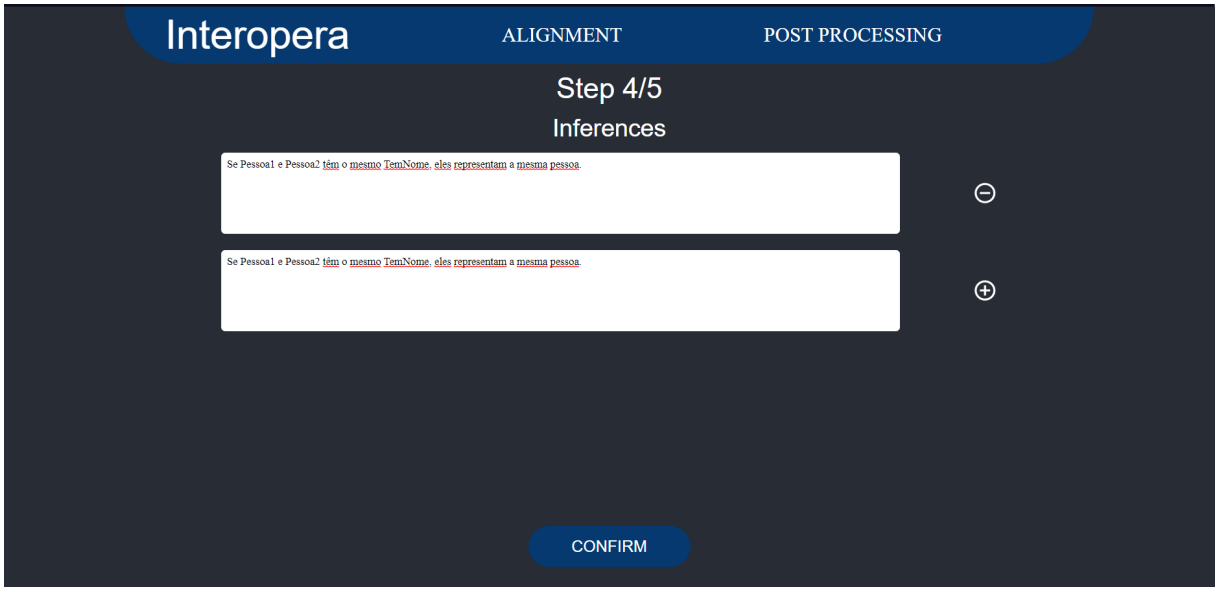


Figura 20: Página de seleção de inferências da interface preenchida

Por fim para começar o processamento dos dados basta clicar no botão "*CONFIRM*", em sequencia a interface será guiada para a tela de processamento enquanto o alinhamento dos dados ocorre, esta tela pode ser vista na Figura 21. Nesta etapa a interface irá aguardar o processamento feito pela ferramenta terminar e em sequencia guiará a interface para a tela de download como pode ser visto na Figura 22.

Por fim, para iniciar o processamento dos dados, basta clicar no botão "*CONFIRM*". Em seguida, a interface será guiada para a tela de processamento, onde ocorrerá o alinhamento dos dados. Essa tela pode ser visualizada na Figura 21. Durante esta etapa, a interface aguardará a conclusão do processamento pela ferramenta e, em seguida, guiará o usuário para a tela de download, conforme ilustrado na Figura 22.

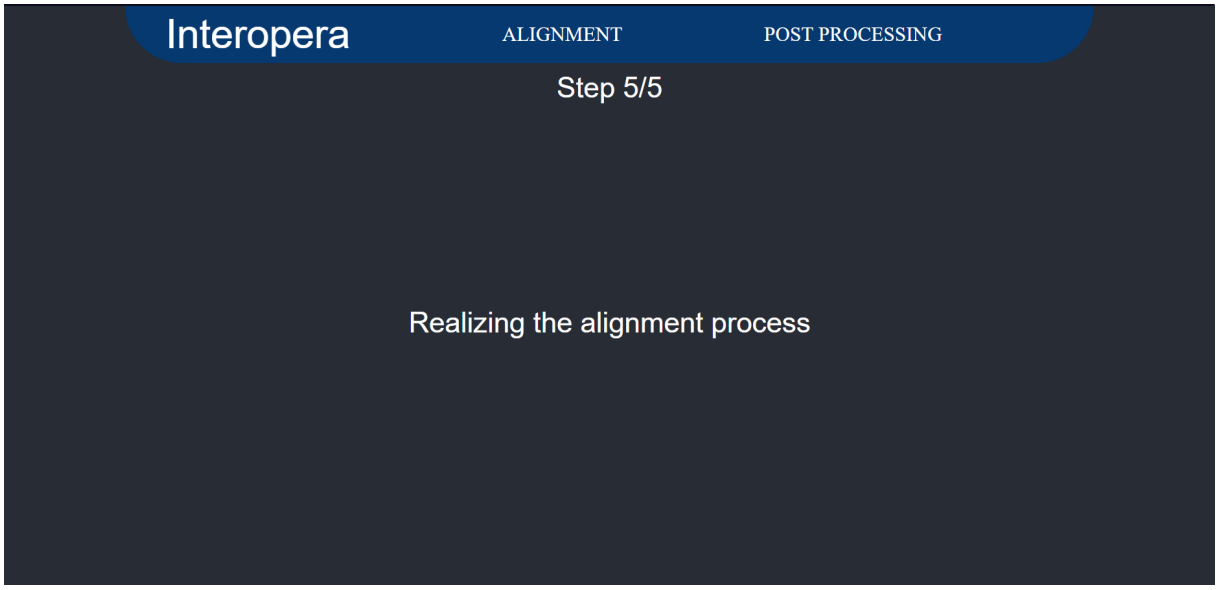


Figura 21: Página de processamento da interface

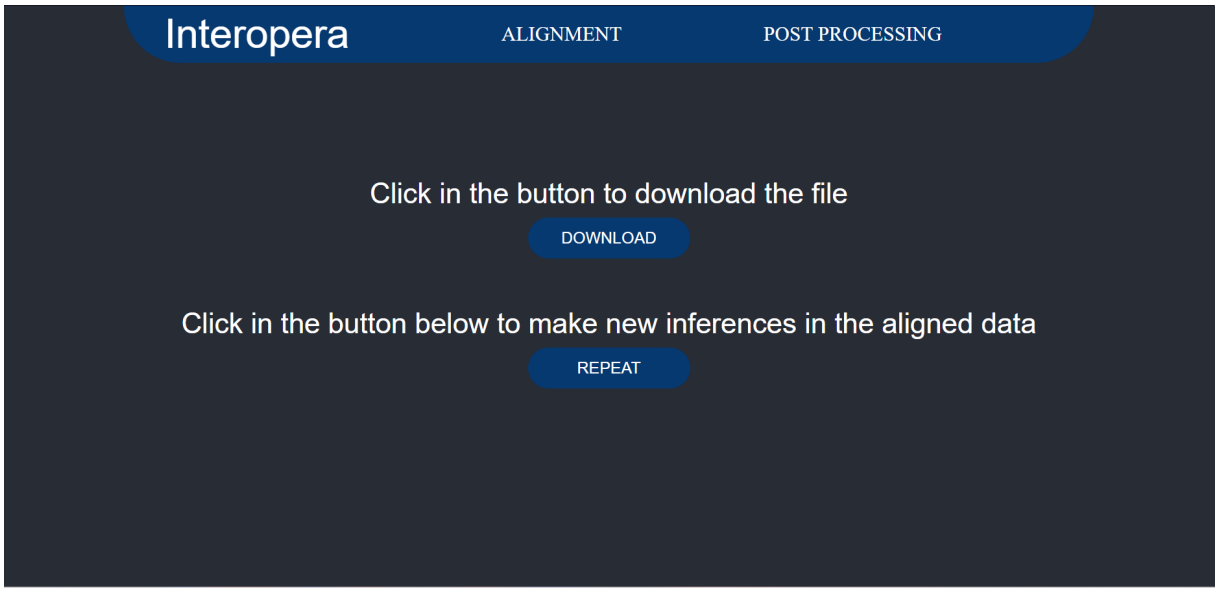


Figura 22: Página de download da interface

Na tela de download, o usuário tem duas opções: baixar os arquivos gerados ou realizar novas inferências sobre os dados. No caso da primeira opção, um arquivo compactado será baixado para o computador do usuário, contendo todos os resultados do processamento. Para a segunda opção, a interface o guiará através de um fluxo de repetição de consultas e inferências, conforme ilustrado nas Figuras 23 e 24.

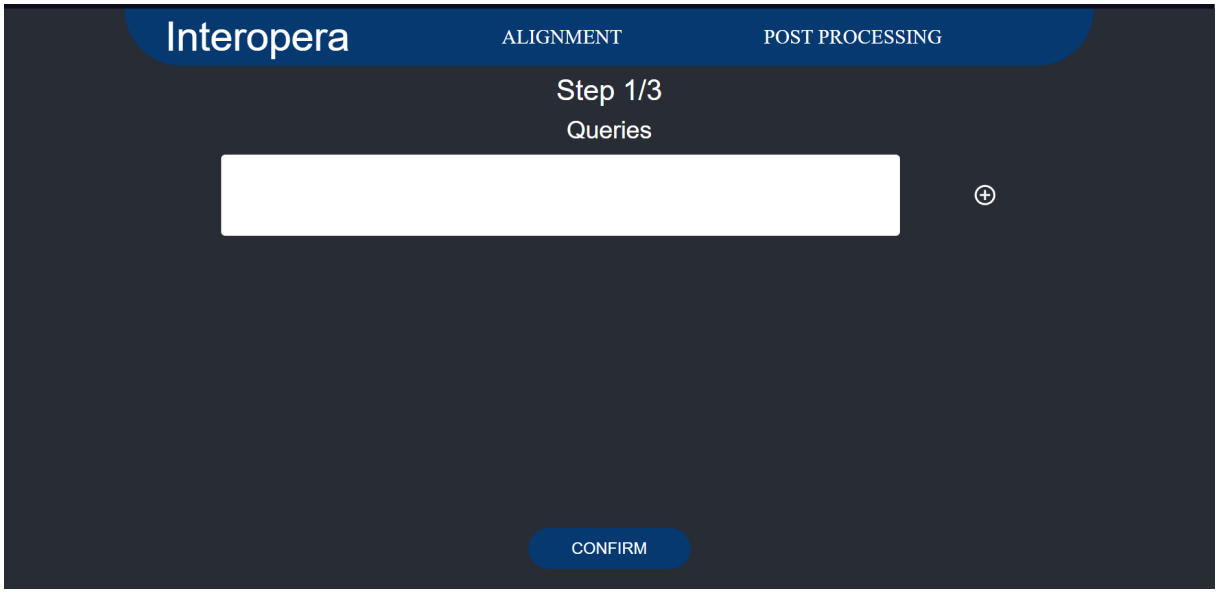


Figura 23: Página de repetição de consultas da interface

Figura 24: Página de repetição de inferências da interface

Por fim, ao adicionar novas inferências ou consultas a serem executadas, a interface exibirá novamente as telas de processamento e download, conforme ilustrado nas Figuras 25 e 26. No entanto, a nova página de downloads conterá apenas os resultados das novas inferências e consultas adicionadas, caso o usuário utilize o botão de baixar os dados. Esta tela permite ao usuário repetir o ciclo de realizar novas inferências até estar satisfeito com os resultados, proporcionando uma experiência mais eficiente e personalizada.

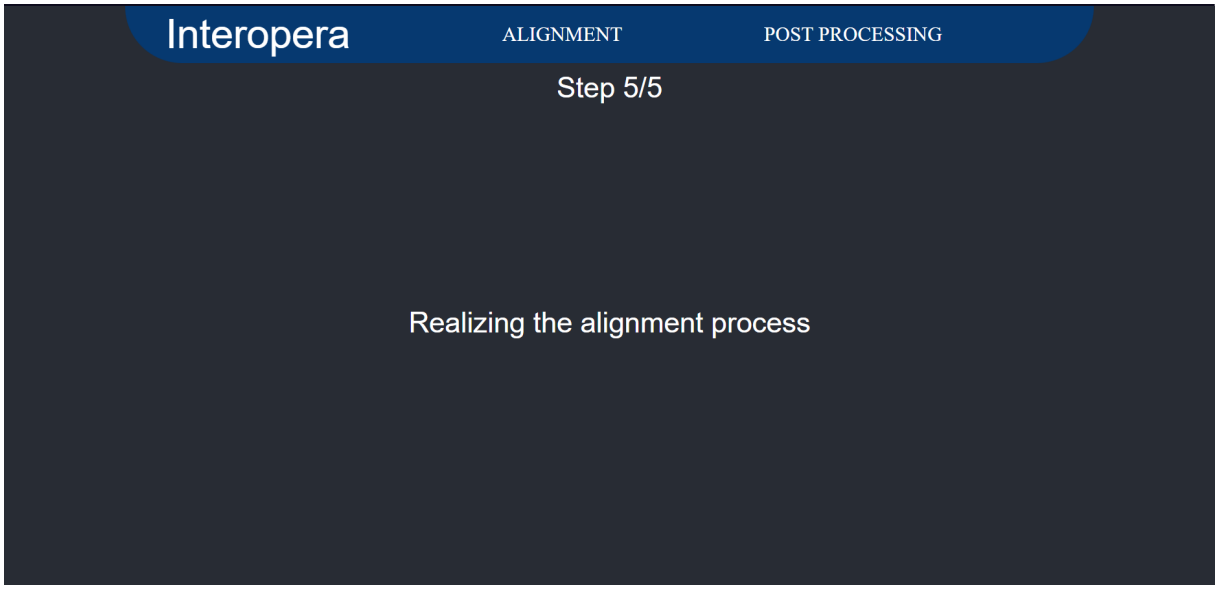


Figura 25: Página de processamento da interface

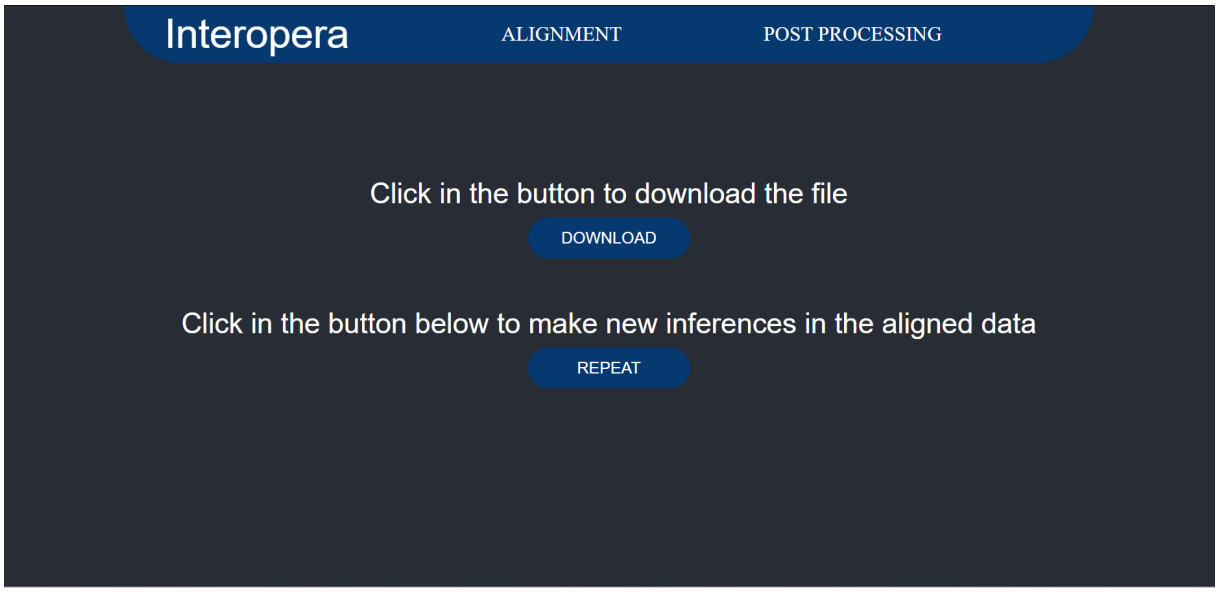


Figura 26: Página de download da interface

Prosseguindo com os fluxos de pós-processamento, a interface exibirá a tela de seleção de processo, como ilustrado na Figura 27, onde serão apresentados os fluxos de Deep Matcher, Pandas e SUMO. Esses fluxos seguem os mesmos procedimentos mencionados no Capítulo 3 e podem ser executados quantas vezes forem desejados. Ao escolher uma das opções e clicar no botão "*CONFIRM*", a interface levará o usuário para a primeira tela do fluxo selecionado, dando início ao processo desejado.

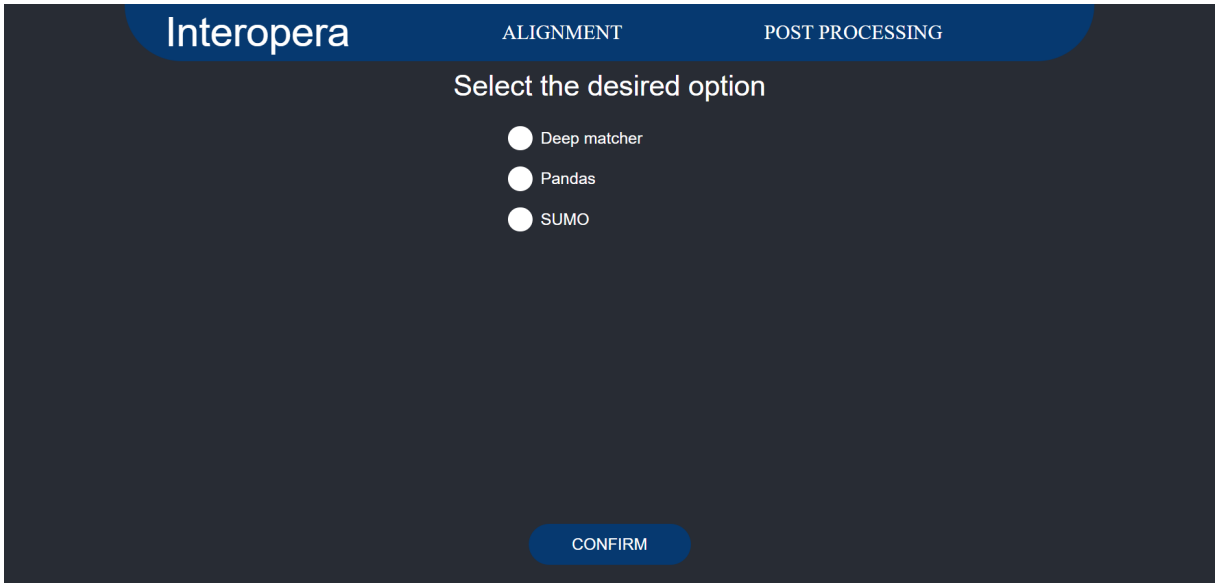


Figura 27: Página de escolha de processo da interface

## 4.1 Deep Matcher

O fluxo de aprimoramento dos dados utilizando o Deep Matcher começa na tela de seleção de arquivos, como ilustrado na Figura 28. Após a seleção dos arquivos, ao clicar no botão "*CONFIRM*", o processo de aprimoramento dos dados terá início, mostrando a tela de aviso de processamento, como ilustrado na Figura 28. Nessa etapa, a interface aguardará até que o processamento realizado pela ferramenta seja concluído. Ao finalizar, a interface guiará o usuário para a tela de download, conforme ilustrado na Figura 30. Nessa tela, basta o usuário clicar no botão de download para baixar os dados aprimorados.

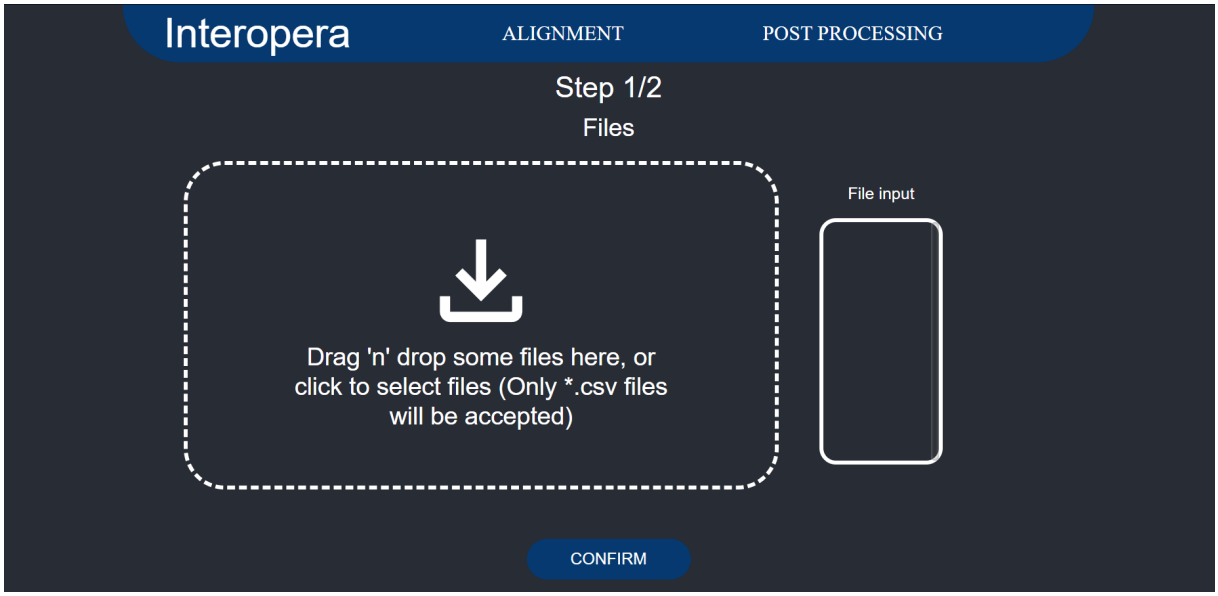


Figura 28: Página de seleção de arquivos do Deep Matcher da interface

Figura 29: Página de processamento do Deep Matcher da interface

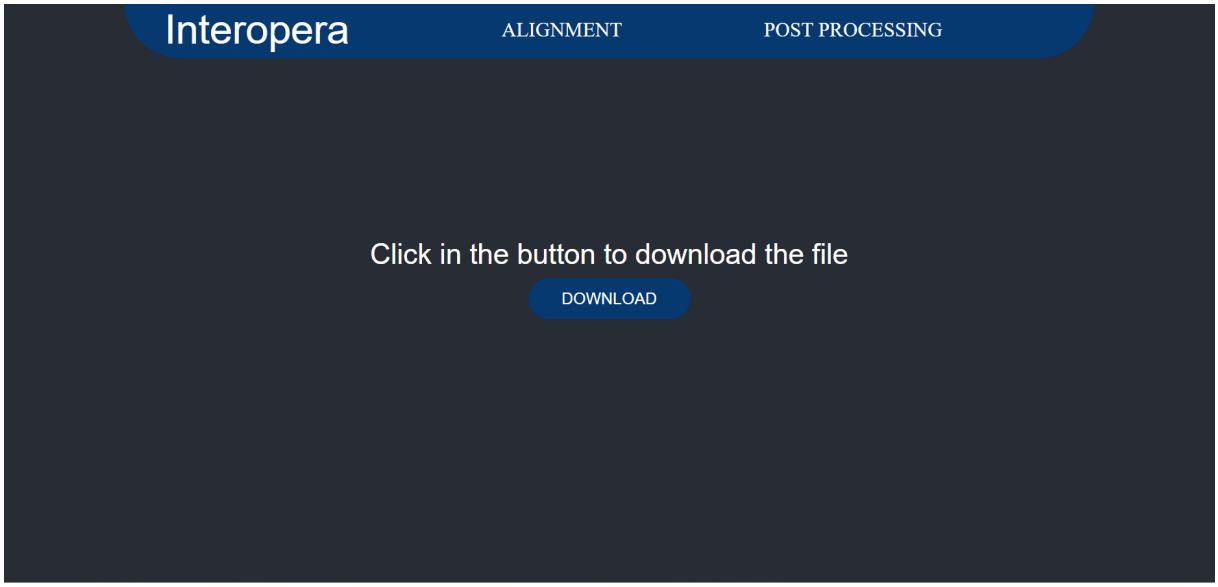


Figura 30: Página de download do Deep Matcher da interface

Por fim, o usuário tem a opção de repetir o processo ao retornar à tela de seleção de processos e executar o fluxo de aprimoramento novamente. Dessa forma, o usuário pode realizar várias iterações de melhoria dos dados, selecionando diferentes processos ou ajustando os parâmetros conforme necessário, proporcionando uma experiência flexível e adaptável às suas necessidades.

## 4.2 Pandas

O fluxo de novas consultas utilizando o Pandas tem início na tela de seleção de arquivos, conforme demonstrado na Figura 31. Nesse ponto, é necessário selecionar apenas um arquivo específico para prosseguir. Após a seleção do arquivo, ao clicar no botão "*CONFIRM*", a interface direcionará o usuário para a tela de seleção de consultas, como ilustrado na Figura 32. Nessa etapa, todas as consultas desejadas devem ser inseridas, e ao finalizar a seleção, o usuário deverá clicar no botão "*CONFIRM*", que guiará a interface para a tela de processamento, como exemplificado na Figura 33.

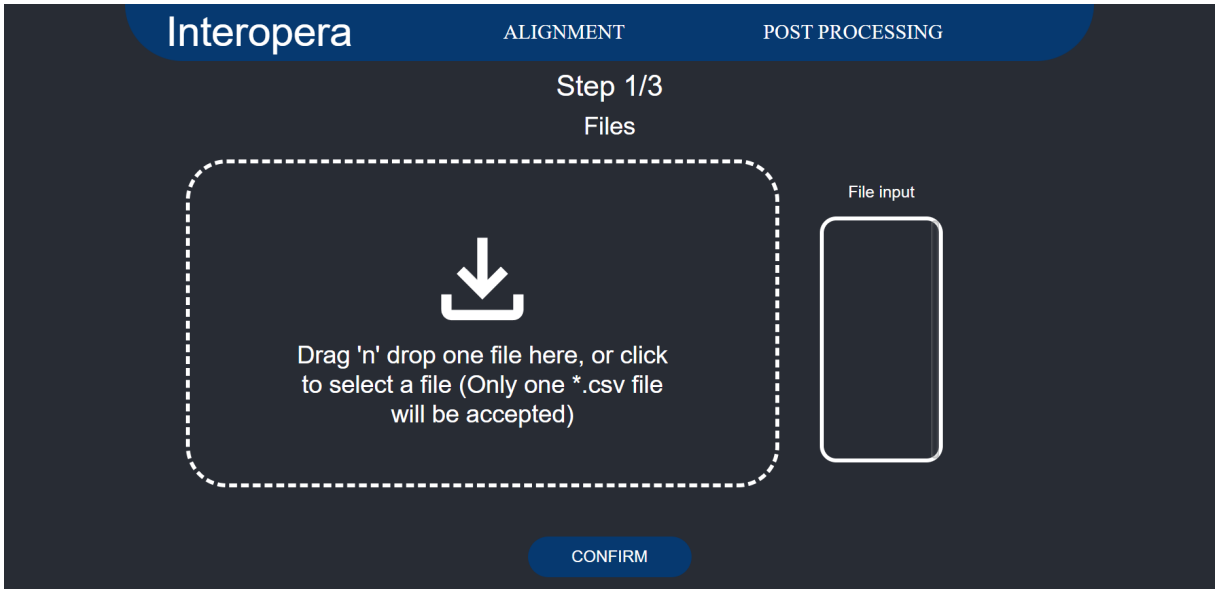


Figura 31: Página de seleção de arquivos do Pandas da interface

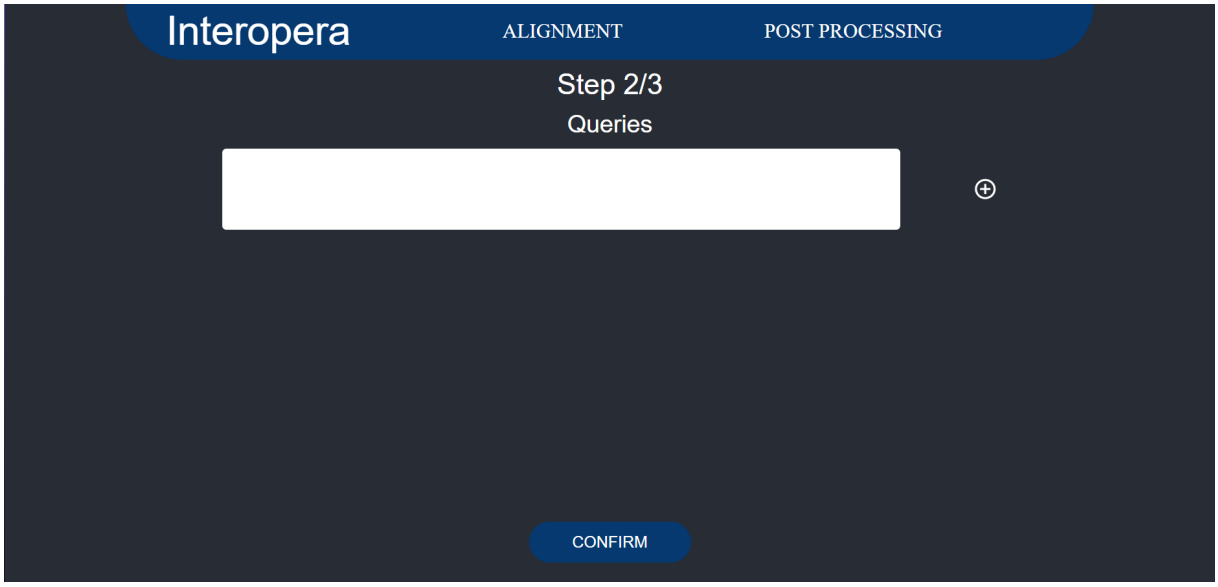


Figura 32: Página de seleção de consultas do Pandas da interface

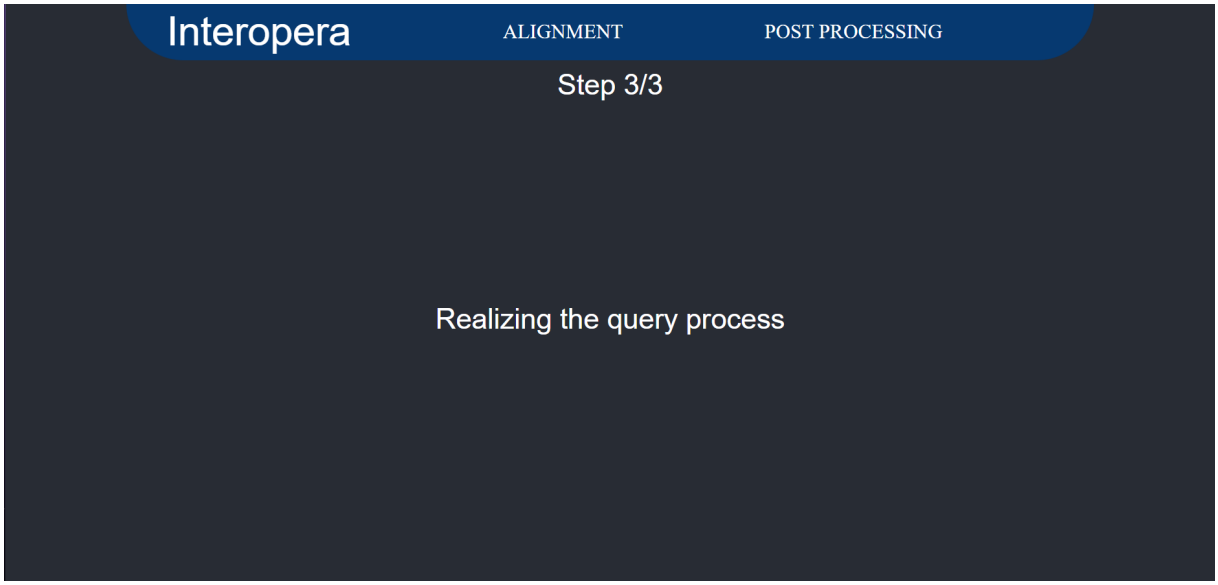


Figura 33: Página de processamento do Pandas da interface

Na tela de processamento, a interface aguardará a conclusão das consultas realizadas pela ferramenta e, uma vez finalizado o processamento, direcionará o usuário para a tela de download, tal como ilustrado na Figura 34. Para baixar os arquivos resultantes das consultas, o usuário deve clicar no botão de "*DOWNLOAD*" e Caso deseje realizar novas consultas, o usuário deverá retornar à tela de seleção de processos e escolher novamente o processo de execução do Pandas.

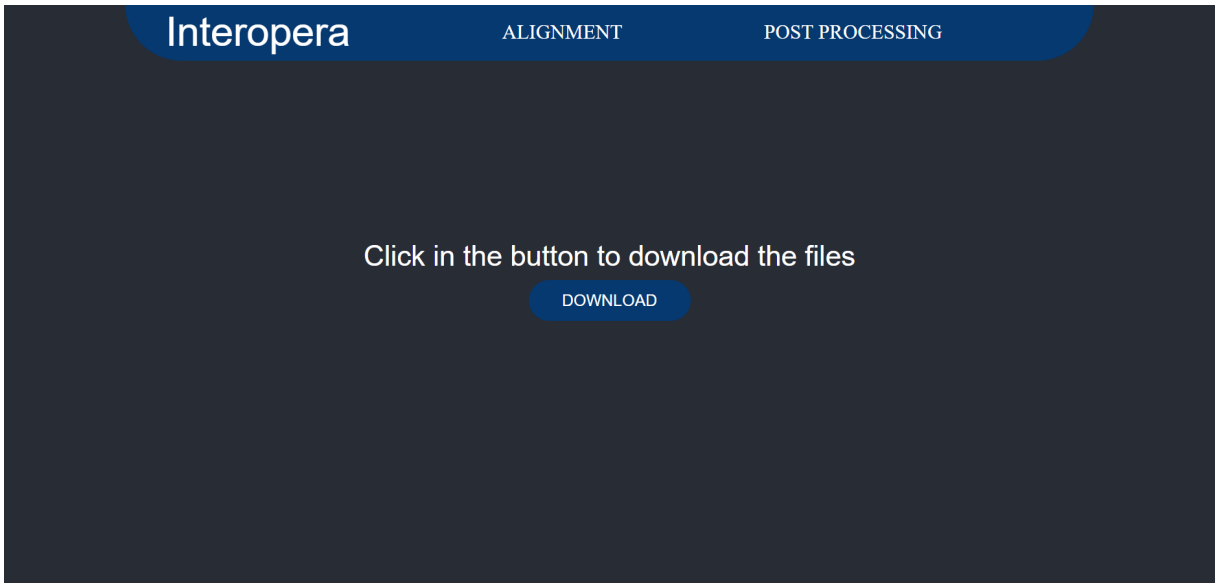


Figura 34: Página de download do Pandas da interface

## 4.3 SUMO

O fluxo de novas inferências utilizando a ontologia SUMO tem início na tela de seleção de arquivos, conforme demonstrado na Figura 35. Nesse ponto, é necessário selecionar apenas um arquivo específico para prosseguir. Após a seleção do arquivo, ao clicar no botão "*CONFIRM*", a interface direcionará o usuário para a tela de seleção de inferências, como ilustrado na Figura 36. Nessa etapa, todas as inferências desejadas devem ser inseridas, e ao finalizar a seleção, o usuário deverá clicar no botão "*CONFIRM*", que guiará a interface para a tela de processamento, como exemplificado na Figura 37.

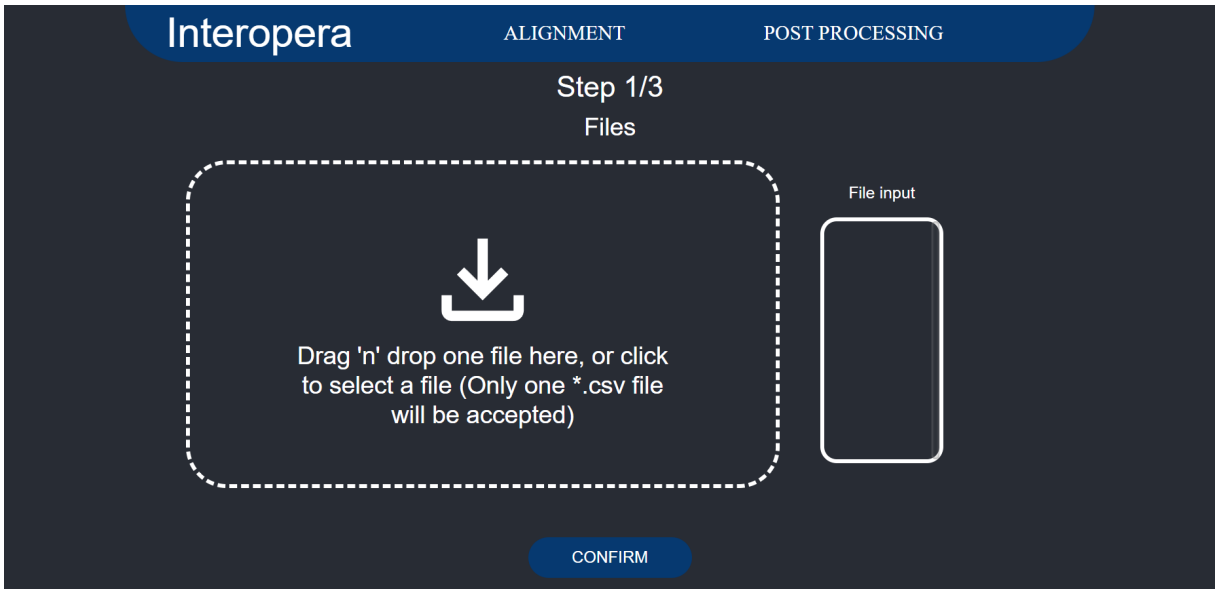


Figura 35: Página de seleção de arquivos do SUMO da interface

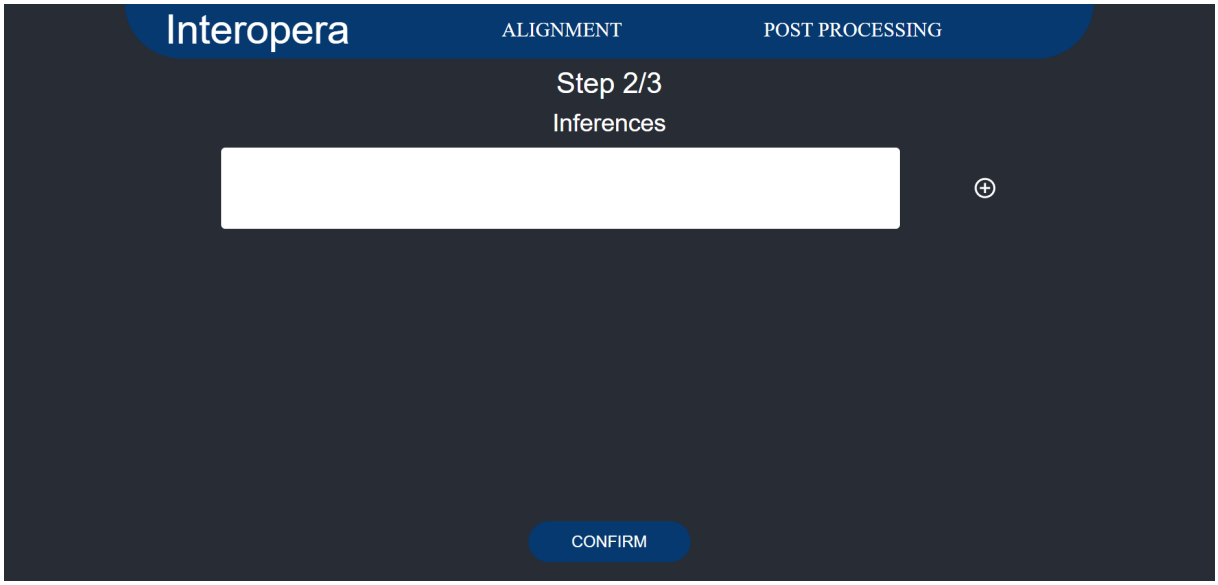


Figura 36: Página de seleção de inferências do SUMO da interface

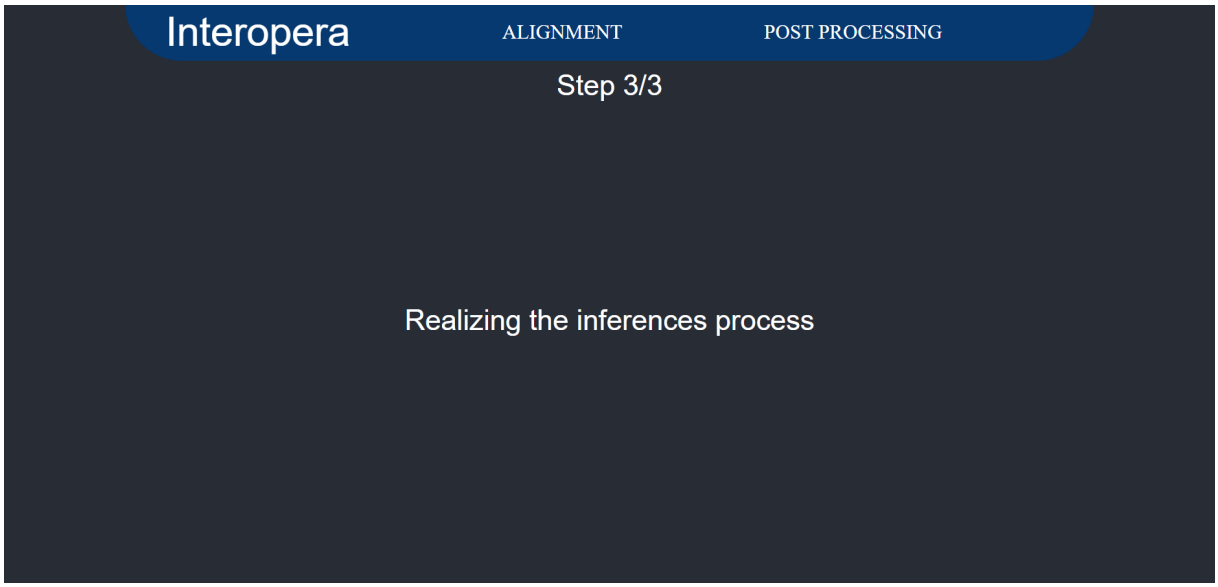


Figura 37: Página de processamento do SUMO da interface

Na tela de processamento, a interface aguardará a conclusão das inferências realizadas pela ferramenta e, uma vez finalizado o processamento, direcionará o usuário para a tela de download, tal como ilustrado na Figura 38. Para baixar os arquivos resultantes das inferências, o usuário deve clicar no botão de "*DOWNLOAD*" e Caso deseje realizar novas inferências, o usuário deverá retornar à tela de seleção de processos e escolher novamente o processo de execução da SUMO.

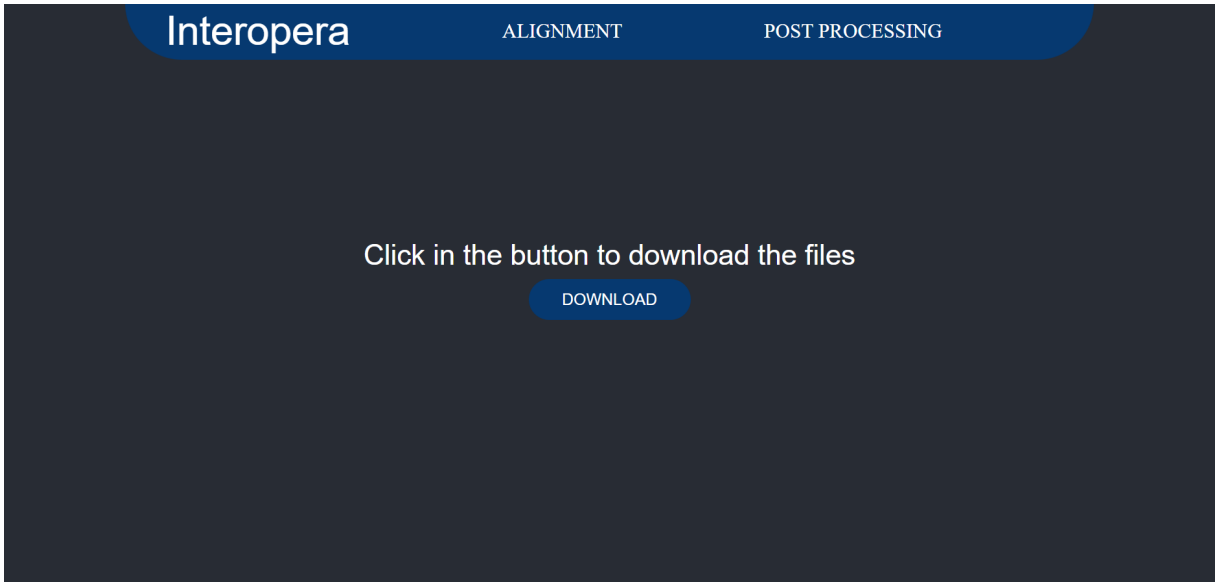


Figura 38: Página de download do SUMO da interface

# 5 Conclusões e trabalhos futuros

Ao longo dos anos, a quantidade de informações disponíveis ao público através de meios digitais tem aumentado consideravelmente. No entanto, esse crescimento ocorre de forma fragmentada, com diversas empresas disponibilizando dados de maneira independente. A integração dessas informações pode resultar em inconsistências e erros no sistema, o que torna esse processo desafiador e dispendioso para as empresas.

Consequentemente, ferramentas que auxiliem no alinhamento de dados têm se tornado cada vez mais requisitadas. Nesse sentido, a arquitetura proposta mostra-se uma opção promissora, uma vez que sua escalabilidade e capacidade de oferecer resultados em múltiplos formatos estão em sintonia com a pluralidade de informações trazida pelo contínuo crescimento dos dados.

Além de realizar os alinhamentos de dados mencionados, a arquitetura tem como propósito enriquecer a expressividade das informações fornecidas pelos usuários, ao sugerir conexões com a ontologia SUMO, que já contém uma vasta gama de dados sobre diversos assuntos. Dessa forma, a arquitetura possibilita um aumento significativo na quantidade de informações disponíveis para os dados recebidos.

Ao incorporar a ontologia SUMO, a ferramenta é capaz de contextualizar e ampliar o entendimento dos dados submetidos pelos usuários, tornando as respostas mais abrangentes e informativas. Essa abordagem impulsiona a qualidade da análise e interpretação dos dados, possibilitando aos usuários explorarem um universo mais amplo de conhecimento.

Com essa sinergia entre a arquitetura e a ontologia SUMO, os usuários podem descobrir novas relações valiosas entre seus dados e o vasto acervo de informações existentes, melhorando assim a tomada de decisões informadas e a descoberta de padrões e correlações que antes poderiam passar despercebidos.

A ferramenta proposta foi validada através de um caso real de alinhamento em um sistema de dados robusto e fragmentado, presente em múltiplos bancos de dados. Além disso, ela oferece aos usuários uma demonstração prática de como iniciar a construção de

uma ferramenta seguindo o modelo da arquitetura.

Durante a validação, a eficácia da ferramenta foi comprovada, proporcionando resultados confiáveis no processo de alinhamento dos dados. Essa abordagem se mostrou particularmente relevante e útil para empresas e organizações que lidam com uma grande quantidade de informações dispersas em diferentes fontes.

Com a utilização dessa arquitetura, os usuários podem contar com uma solução prática e eficiente para enfrentar os desafios de integração de dados em ambientes complexos. Ao seguir o modelo da ferramenta, novos projetos e sistemas podem ser construídos de forma mais estruturada, garantindo uma melhor organização e aproveitamento das informações disponíveis.

Dessa forma, a ferramenta não apenas oferece resultados valiosos no presente, mas também abre caminho para o desenvolvimento contínuo de soluções alinhadas com as demandas crescentes da gestão de dados no cenário atual.

A interface proposta foi desenvolvida com o objetivo de tornar a ferramenta acessível a um grupo maior de usuários e facilitar sua utilização. Com suas características focadas em ser intuitiva e amigável, a interface proporciona uma experiência de uso simplificada, permitindo que mesmo usuários com pouca experiência técnica possam se beneficiar plenamente da ferramenta.

A abordagem centrada no usuário foi uma prioridade durante o desenvolvimento, resultando em uma interface que oferece clareza e praticidade para realizar o alinhamento de dados de forma mais simples pelos usuários.

Dessa forma, a disponibilização da ferramenta por meio de uma interface amigável abre novas possibilidades para uma ampla gama de usuários, permitindo que mais pessoas possam aproveitar os benefícios da ferramenta de forma descomplicada e produtiva.

Os próximos passos desta dissertação se concentram em várias áreas estratégicas para aprimorar a ferramenta e arquitetura. Essas áreas abrangem desde a exploração de novas opções de alinhamento até o estudo de meios adicionais para enriquecer os dados do usuário. Um destes passos envolve a expansão das funcionalidades da ferramenta. Isso incluirá a adição de novos serviços, ampliando assim a gama de funcionalidades oferecidas para os usuários.

Outro foco importante é a melhoria da responsividade da interface da ferramenta. Esse aprimoramento garantirá que a interface funcione de maneira eficaz em diferentes dispositivos, tornando-a acessível e amigável para um público mais amplo. O aumento

da integração com a ontologia SUMO pode ser aprimorado, visando enriquecer ainda mais os resultados fornecidos pela ferramenta. Essa integração permitirá que os usuários obtenham informações mais abrangentes e contextuais.

Outros passos incluem tornar a interface da ferramenta disponível online. Essa medida visa aprimorar o acesso, tornando-o conveniente e remoto para atender às necessidades de um número maior de usuários. A incorporação de dados de experimentos, utilizando, por exemplo, informações de empresas públicas de grande relevância. A verificação de possíveis integrações com outras ontologias ou bases de dados relevantes para aprimorar ainda mais a qualidade e a profundidade das análises. Realizar mais testes de desempenho da ferramenta em cenários que envolvam grandes volumes de dados. Essa avaliação é crucial para garantir que a ferramenta funcione eficazmente, independentemente do tamanho da base de dados.

# REFERÊNCIAS


AHO, A.V.; ULLMAN, J.D. **Foundations of Computer Science: C Edition**. [S. l.]: W. H. Freeman, 1994. (Principles of computer science series). ISBN 9780716782841. Disponível em: <https://books.google.com.br/books?id=q7-HQgAACAAJ>.

BARRIÈRE, C. **Natural Language Understanding in a Semantic Web Context**. [S. l.]: Springer International Publishing, 2016. ISBN 9783319413372. Disponível em: <https://books.google.com.br/books?id=szaIDQAAQBAJ>.

BIRD et al. **Natural Language Processing with Python**. [S. l.]: O'Reilly Media Inc, 2009.

CANDELA, Leonardo; CASTELLI, Donatella; PAGANO, Pasquale. Data Interoperability. **Data Science Journal**, v. 12, jul. 2013. DOI: 10.2481/dsj.GRDI-004.

FELLBAUM, Christiane. **WordNet: An Electronic Lexical Database**. [S. l.]: The MIT Press, mai. 1998. ISBN 9780262272551. DOI: 10.7551/mitpress/7287.001.0001. Disponível em: <https://doi.org/10.7551/mitpress/7287.001.0001>.

FRANZ et al. **The Description Logic Handbook: Theory, Implementation, and Applications**. [S. l.]: Cambridge University Press, 2nd Edition, 2007.

LUCIANO; FLORIDI. **Blackwell Guide to the Philosophy of Computing and Information.** [S. l.]: Oxford: Blackwel, 2003.

MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In_____. **Proceedings of the 9th Python in Science Conference**. [S. l.: s. n.], 2010. P. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

MILLER, George A. WordNet: A Lexical Database for English. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 11, p. 39–41, nov. 1995. ISSN 0001-0782. DOI: 10.1145/219717.219748. Disponível em: <https://doi.org/10.1145/219717.219748>.

NILES, Ian; PEASE, Adam. Towards a Standard Upper Ontology. In: PROCEEDINGS of the International Conference on Formal Ontology in Information Systems - Volume 2001. Ogunquit, Maine, USA: Association for Computing Machinery, 2001. (FOIS '01), p. 2–9. ISBN 1581133774. DOI: 10.1145/505168.505170. Disponível em: <https://doi.org/10.1145/505168.505170>.

REBACK, Jeff et al. **pandas-dev/pandas: Pandas 1.4.0**. [S. l.]: Zenodo, jan. 2022. DOI: 10.5281/zenodo.5893288. Disponível em: <https://doi.org/10.5281/zenodo.5893288>.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S. l.]: Pearson, 2016. (Always learning). ISBN 9781292153964. Disponível em: <https://books.google.com.br/books?id=XS9CjwEACAAJ>.

[S. l.].SIGMOD '18: Proceedings of the 2018 International Conference on Management of Data. Houston, TX, USA: Association for Computing Machinery, 2018. ISBN 9781450347037.

SUMO, Suggested Upper Merged Ontology. Disponível em: <https://www.ontologyportal.org/index.html>.

SZEREDI, P. et al. **The Semantic Web Explained: The Technology and Mathematics behind Web 3.0**. [S. l.]: Cambridge University Press, 2014. ISBN 9780521700368. Disponível em: <https://books.google.com.br/books?id=bvwOBAAAQBAJ>.

TEXTDISTANCE, python library. [S. l.: s. n.]. https://pythonrepo.com/repo/orsinium-textdistance-python-text-processing.

UNIVERSITY, Princeton. **WordNet**. 2023. Disponível em: <https://wordnet.princeton.edu/>.

WU, Yonghui et al. **Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation**. [S. l.: s. n.], 2016. arXiv: 1609.08144 [cs.CL].