Universidade Estadual de Londrina

WILLIAM G. R. MEDINA

AVALIAÇÃO DE FERRAMENTAS DO ESTADO DA ARTE PARA DETECÇÃO DE ERROS EM BANCOS DE DADOS RELACIONAIS

LONDRINA
2022

WILLIAM G. R. MEDINA

AVALIAÇÃO DE FERRAMENTAS DO ESTADO DA ARTE PARA DETECÇÃO DE ERROS EM BANCOS DE DADOS RELACIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Daniel S. Kaster

Coorientador: Prof(a). Dr. Eduardo H. M. Pena

LONDRINA

2022

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

WILLIAM G. R. MEDINA

AVALIAÇÃO DE FERRAMENTAS DO ESTADO DA ARTE PARA DETECÇÃO DE ERROS EM BANCOS DE DADOS RELACIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Daniel S. Kaster
Universidade Estadual de Londrina

Prof. Dr. Segundo Membro da Banca
Universidade/Instituição do Segundo
Membro da Banca – Sigla instituição

Prof. Dr. Terceiro Membro da Banca
Universidade/Instituição do Terceiro
Membro da Banca – Sigla instituição

Prof. Ms. Quarto Membro da Banca
Universidade/Instituição do Quarto
Membro da Banca – Sigla instituição

Londrina, 27 de junho de 2022.

Este trabalho é dedicado aos meus familiares e amigos, que trilharam esta jornada comigo.

AGRADECIMENTOS

Em primeiro lugar a Deus que me põe de pé e me guia a cada dia, me trazendo paz, saúde e tranquilidade, mesmo em tempos atípicos de pandemia. Ao corpo docente da Universidade Estadual de Londrina e ao Departamento de Computação, assim como ao pessoal de administração. Aos professores Daniel dos Santos Kaster e Eduardo H. M. Pena, orientador e coorientador, respectivamente, pelo conhecimento, dedicação e ajuda para a realização deste trabalho. Agradeço a minha família e amigos por estarem ao meu lado durante esta etapa final da conclusão do meu curso.

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

MEDINA, W. G. R. **AVALIAÇÃO DE FERRAMENTAS DO ESTADO DA ARTE PARA DETECÇÃO DE ERROS EM BANCOS DE DADOS RELACIONAIS**. 2022. 49f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

A coleta de dados frequentemente introduz erros acidentais. Células vazias, grafias ou digitações errôneas, violações de regras de integridade e de negócios e duplicações involuntárias de informações são comuns e podem ter severos impactos negativos em tarefas de análises de dados. Com a popularização da área de *Data Science*, a curadoria, unificação e limpeza de dados têm se mostrado essenciais para extrair o máximo valor em levantamentos de qualquer natureza, evitando inconsistências e garantindo a manipulação esperada. Vários métodos e sistemas de detecção de erros se propuseram a resolver os desafios intrínsecos à correção de dados. Muitos destes métodos surgiram nos últimos anos, introduzindo dificuldades na operação e identificação de qual ferramenta é melhor ou pior para determinados tipos de erro. Este trabalho propõe a avaliação dos diferentes métodos e ferramentas disponíveis ao público geral levando-se em conta variados cenários, trazendo uma combinação dos métodos do estado da arte. Através da introdução controlada de erros em conjuntos de dados, foi possível avaliar o efeito que erros de diversos tipos causam a cada algoritmo de limpeza de dados. Os resultados obtidos demonstram que ferramentas baseadas em aprendizado de máquina possuem capacidade superior de detecção de erros em relação a métodos anteriores, mas só obtêm vantagem significativa quando a porcentagem de erros é relativamente alta (na ordem de 5%), ou quando as tuplas dos conjuntos de dados são pouco distintas. As conclusões também mostraram que ferramentas que utilizam técnicas de agrupamento de dados, como o Trifacta Data Wrangler, não são necessariamente alternativas ruins e funcionam bem quando o conjunto de dados é composto exclusivamente por erros simples como erros de digitação. Por outro lado, essas ferramentas parecem encontrar obstáculos significativos quando o tipo de erro inserido é estrutural.

Palavras-chave: Limpeza de dados. Detecção de erros. Correção de erros. Perfil de dados. Manipulação de dados. Dados relacionais.

MEDINA, W. G. R. **ASSESSMENT OF STATE OF THE ART TOOLS FOR DETECTION AND CORRECTION OF ERRORS IN DATA RECORDS**. 2022. 49p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

Data collection often introduces accidental errors. Empty cells, misspellings or erroneous data, integrity and constraint violations as well as duplicate data are surprisingly common and can produce severe negative impacts on data analysis tasks. With the popularization of Data Science, the curation, unification, preparation and cleaning of data has proven to be essential in order to extract maximum value from data queries, avoiding inconsistencies and guaranteeing expected results. Numerous tools and systems have been proposed to fix these issues. Many of these have emerged in recent years, introducing a range of difficulties in operating and identifying the error detection capacity of each tool in different contexts. This work proposes the evaluation of each method taking into account different scenarios, bringing together a combination of methods from recent literature. By introducing specific types of errors in specific sets of data, it was possible to identify the effect caused by variations in the percentage of certain types of errors on the performance of data cleaning algorithms. The results obtained demonstrate that tools based on machine learning have superior error detection capabilities compared to previous methods, but only obtain a significant advantage when the percentage of errors is relatively high. Conclusions also showed that tools that use data clustering techniques, like Trifacta Data Wrangler, are not necessarily bad alternatives and work well when the dataset is composed exclusively by simple errors like typos. On the other hand, these tools seem to encounter significant obstacles when the type of error inserted is structural.

Keywords: Data cleaning. Error detection. Error correction. Data profiling. Data wrangling. Relational data.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
BART	*Benchmarking Algorithms for (data) Repairing and (data) Translation*
DFKI	*Deutsches Forschungszentrum für Künstliche Intelligenz*
DCs	*Denial Constraints*
SGBD	Sistema Gerenciador de Banco de Dados
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*

SUMÁRIO

1 INTRODUÇÃO

Empresas e instituições em geral vêm coletando grandes quantidades de dados de uma variedade de fontes complementares. Os repositórios de dados gerados têm como objetivo final alimentar tarefas analíticas, possibilitando integrar dados sob diferentes aspectos para gerar informações com valor agregado para o propósito da organização. A manipulação de diferentes fontes de dados frequentemente introduz erros que podem causar problemas em projetos analíticos, particularmente naqueles que se utilizam de técnicas de aprendizado de máquina. Krishman et. al [2] apontam que a corrupção sistemática em uma só variável é o suficiente para causar modelos deslocados de *machine learning*, e treinar com dados errados podem trazer resultados ainda piores de aprendizado do que utilizar execuções sem treinamento algum. Por conta destes problemas, a curadoria de dados têm se mostrado cada vez mais em pauta para tarefas analíticas.

Com o aumento do interesse na área de ciência de dados e o crescimento do volume de dados armazenados por empresas, cresceu também o tempo atribuído a tarefas de análise. Uma pesquisa com cientistas de dados feita pela companhia americana de Inteligência Artificial (IA) CrowdFlower (atualmente denominada Figure Eight) e publicada no website da revista Forbes [1] exibiu que mais de 60% do tempo investido em projetos de ciência de dados é destinado à limpeza e organização de dados. Além deste fator, 57% dos cientistas de dados alegaram que a limpeza e organização dos dados é a parte menos agradável do seu trabalho.

Limpeza de dados tipicamente compreende duas fases: (1) detecção de erros, onde erros de tipos variados são identificados, e (2) reparo de erros, em que atualizações são aplicadas aos dados, automaticamente ou por sugestão de usuários especialistas, gerando uma versão mais "limpa" do conjunto de dados [1]. Técnicas de detecção de erros podem ser quantitativas ou qualitativas [1]. Técnicas quantitativas frequentemente utilizam métodos estatísticos para identificar valores anormais ou extremos, que diferem muito do comportamento do restante do conjunto de dados. Por exemplo, um resultado de exame laboratorial com valor muito acima dos valores máximos tipicamente identificados deve ser um erro. Já técnicas qualitativas de detecção de erros baseiam-se em abordagens que identificam padrões ou restrições de integridade válidas para uma instância consistente de um conjunto de dados e reportam como erros os valores que violam tais padrões ou restrições [1].

Nos últimos anos, várias ferramentas comerciais de limpeza de dados foram desenvolvidas. No entanto, grande parte destas ferramentas se limita a métodos que exigem

[1] <https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>

algum grau de conhecimento do usuário a respeito de regras de dependência e de negócio referentes ao conjunto em análise, ou apresentam grau de detecção de erros insatisfatório [4].

Um estudo de Abedjan et. al [4] propôs a comparação unificada do desempenho de diferentes ferramentas e algoritmos de limpeza de dados com alguns conjuntos de dados reais. No entanto, desde o estudo de Adebdjan et. al, propostas com a utilização de algoritmos de aprendizado de máquina têm sido apresentadas e, em boa parte dos casos, têm sustentado argumentos expressivos em tarefas de limpeza de dados. Por exemplo, o HoloDetect [5] apresenta a ideia de um algoritmo que, utilizando-se de técnicas de *few-shot learning* baseadas em supervisão fraca [6, 7], demonstra taxas de precisão e revocação (seção 2.2) de 0,903 e 0,989, respectivamente, para o conjunto de dados Hospital[2]. Em comparação, algoritmos tradicionais baseados em detecção de *outliers* - amplamente empregadas em ferramentas como o DBoost [3] - apresentam taxas de 0,64 e 0,667 para as mesmas métricas e o mesmo conjunto de dados.

Apesar do avanço na área de limpeza de dados trazer consigo uma série de soluções computacionais para problemas relacionados a erros em dados relacionais [5, 8, 9], grande parte destas propostas divergem em seus estudos experimentais e dificultam ao analista de dados decidir qual solução se enquadra melhor ao seu problema, uma vez que o tipo e quantidade de erros introduzidos são apenas alguns dentre vários fatores que podem gerar alta variação no que tange ao desempenho e capacidade de detecção de erros de cada ferramenta [4].

O objetivo deste trabalho foi a análise experimental e atualizada do estado da arte em detecção automática de erros, sistematizando e mensurando o potencial de cada ferramenta para o campo de limpeza de dados com conjuntos variados de dados e um ambiente unificado de testes. Com a introdução controlada de erros em conjuntos específicos de dados, foi possível identificar o comportamento de cada ferramenta disponível quando submetidas a variações específicas como a mudança de taxas de erros ou a alteração no tipo de erro introduzido. Os resultados obtidos desmonstram que ferramentas que se utilizam de meios de aprendizado de máquina possuem capacidade superior de detecção de erros em relação a métodos anteriores, mas só obtêm vantagem significativa quando a porcentagem de erros é relativamente alta. Foi possível também identificar que ferramentas de aprendizado de máquina utilizam quantidades significativas de memória computacional. Por isso, estão limitadas a conjuntos menores de dados ou necessitam de concessões computacionais que podem afetar consideravelmente o desempenho da detecção de erros em dados.

Este trabalho possibilitou uma visão não somente atualizada, mas também diferente de estudos anteriores, visto que estudos passados limitam-se a análises de desempe-

[2] <https://github.com/HoloClean/holoclean/blob/master/testdata/hospital.csv>

nho em conjuntos de dados distintos e, portanto, impossibilitam mapeamentos precisos do efeito de mudanças específicas em conjuntos de dados.

Este documento está organizado da seguinte forma. O capítulo 2 apresenta fundamentos teóricos necessários para a compreensão da área de limpeza de dados e discorre sobre estudos relacionados à área de ciência de dados. O capítulo 3 apresenta uma proposta de solução para as carências encontradas em estudos atuais na área de limpeza de dados e fornece detalhamentos sobre o experimento realizado. São detalhados escolhas e configurações de ferramentas utilizadas, conjuntos de dados escolhidos e parametrizações para cada teste realizado. O capítulo 4 descreve os resultados obtidos a partir dos experimentos. Por fim, o capítulo 5 descreve conclusões e observações sobre possíveis estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

2.1 Limpeza de dados

A limpeza de dados consiste na implementação de estratégias para corrigir erros, garantindo integridade da informação em bancos de dados relacionais. A Figura 1 indica o típico fluxo de uma tarefa de limpeza de dados.

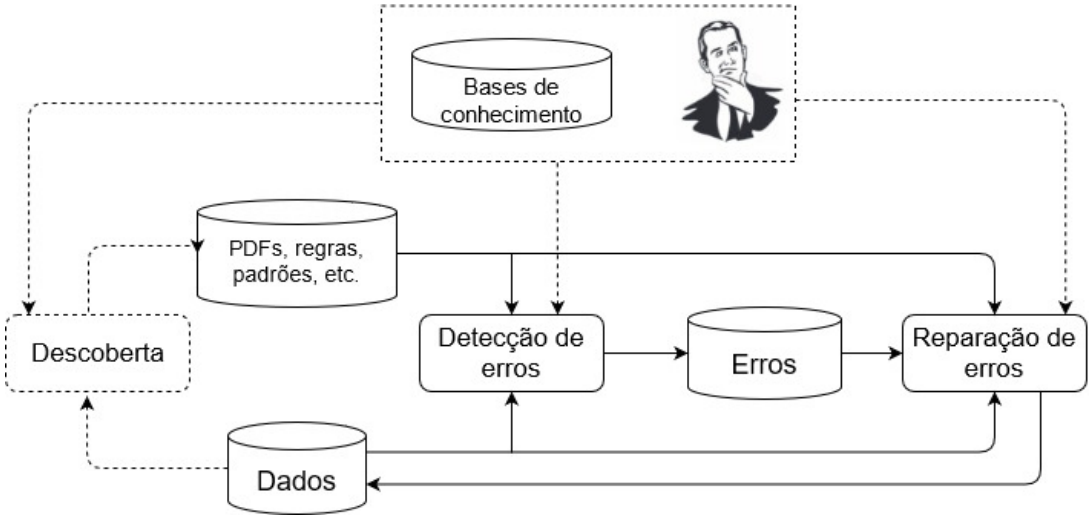

Figura 1 – Tarefa de limpeza de dados. (Fonte: [1], tradução nossa.)

Concentrando-se na etapa de detecção de erros, dado um conjunto de dados, varreduras são executadas de forma a identificar erros. Uma proposta simples é a consulta a um especialista em domínio que, partindo de bases de conhecimento, consegue identificar problemas. Por se tratar de um método manual de detecção, exige tempo, consome recursos e tem relação unívoca com o tamanho do conjunto analisado. Além disso, para que haja uma detecção de erros eficiente é necessário que se conheça todos os metadados referentes ao conjunto.

Metadados referem-se a "dados sobre dados" [1]. Suponha que haja um conjunto de dados que guarde informações de todos os CEPs do Brasil e seus respectivos estados. Em

[1] <https://www.merriam-webster.com/dictionary/metadata>

uma análise cuidadosa, nota-se que todos os CEPs do estado do Paraná estão na faixa de 80000- 000 a 87999-999. Nesse sentido, a afirmação "todos os CEPs do Paraná estão na faixa de 80000-000 a 87999-999" pode ser considerada como metadado sobre os CEPs do Brasil.

A etapa responsável pela revelação de regras e metadados sobre o conjunto analisado é representada pelo passo de descoberta.

Uma vez que se conheça informações sobre metadados de um conjunto e/ou se detectem erros, é possível prosseguir à etapa de confirmação e reparação. No caso de detecções automáticas de erros, há a acusação dos mesmos quando uma informação não condiz com seu respectivo metadado, ou estima-se que a informação esteja incorreta. Através da utilização de técnicas de limpeza de dados, tal como a detecção de *outliers* [10], isto é, valores muito discrepantes quando comparados a seus similares, é possível a identificação de valores que não correspondem ao esperado.

Um dos grandes problemas a serem contornados na área de limpeza de dados automatizada ou semiautomatizada é a capacidade de detecção de erros. Em uma análise perfeita, espera-se que as taxas de precisão e de revocação (seção 2.2) estejam ambas em 1, indicando que a cada cem erros detectados, cem destes são realmente erros, e a cada cem erros presentes no conjunto de dados, todos os cem foram detectados. Porém, Abedjan et. al [4] mostram que, mesmo com a combinação de várias ferramentas, taxas de precisão e revocação tão baixas quanto 12,8% e 57,5% foram identificadas quando testando a atuação de ferramentas de limpeza de dados no conjunto de dados real "Animal" [4], indicando um longo caminho a ser percorrido para taxas de desempenho satisfatórias. Outros estudos [8, 5] chegam a apontar taxas ainda menores, dependendo da ferramenta de detecção e o conjunto de dados utilizado. Testes feitos na publicação do Raha [8] demonstraram que a ferramenta ActiveClean[2], por exemplo, obteve taxas de precisão e revocação de 0,02 e 0,01 para o conjunto de dados Movies, do respositório Magellan.

2.2 Métricas de Avaliação

Existem três métricas muito utilizadas para avaliar a eficiência de algoritmos de limpeza de dados: precisão, revocação e medida F1 [5, 3, 8, 11, 9].

2.2.1 Precisão

Precisão é uma métrica que avalia a proporção de identificações que estão realmente corretas dentre um conjunto apontado como correto. Formalmente, ela pode ser definida pela fórmula

$$\frac{VP}{VP + FP} \tag{2.1}$$

onde VP é a quantidade de verdadeiros positivos do conjunto e FP é a quantidade de falsos positivos do conjunto. Uma precisão de 0,33 indica que a cada 100 células apontadas como erro, apenas 33 são realmente erros.

2.2.2 Revocação

Revocação é uma métrica que avalia a proporção de identificações corretas dentre todas as corretas possíveis. Matematicamente, a taxa de revocação pode ser definida pela fórmula

$$\frac{VP}{VP + FN} \tag{2.2}$$

onde VP é a quantidade de verdadeiros positivos do conjunto e FN é a quantidade de falsos negativos do conjunto. Uma taxa de revocação de 58% indica que dentre 100 erros em células, apenas 58 destes são detectados.

2.2.3 Medida F1

Por fim, a taxa de medida F1 é uma métrica que avalia a acurácia de um teste. Ela é calculada a partir da precisão e taxa de revocação de um teste, sendo a média harmônica de ambas. Matematicamente, ela é definida pela fórmula

$$2 \cdot \frac{P \cdot R}{P + R} \tag{2.3}$$

onde P é a taxa de precisão e R é a taxa de revocação.

2.3 Erros em dados

Não importa qual seja o caso de uso exato para seus dados, a qualidade dos dados é importante. Sem ele, os dados não podem cumprir a finalidade pretendida. Erros em um banco de dados de endereços impediriam você de usar os dados para alcançar os clientes de forma eficaz. Um banco de dados de números de telefone que nem sempre inclui códigos de área para cada entrada não fornece as informações necessárias para usar os dados em muitas situações.

Erros em dados tipicamente são compostos por erros de coluna única, erros de dependência ou erros de dados não relacionais. A grande maioria dos estudos em limpeza de dados não utilizam conjuntos de dados não relacionais. Portanto, as duas primeiras categorias serão aqui abordadas.

2.3.1 Erros de coluna única

Erros de coluna única referem-se àqueles que não possuem dependências funcionais com outras colunas. Valores duplicados em uma coluna com restrição de unicidade

ou valores vazios em colunas de preenchimento obrigatório são exemplos de erros de coluna única, pois independem de outras colunas para serem identificados como erros. Estes equívocos apresentam um grande desafio para analistas de dados, principalmente àqueles que utilizam de modelos baseados em aprendizado de máquina, pois podem gerar modelos distorcidos [2]. A Figura 2 indica os efeitos de análises de aprendizado de máquina em modelos com dados limpos (valores corretos) e sujos (valores incorretos mas que foram erroneamente tratados como verdade), sob diversas configurações. A linha azul (contínua) indica a curva de aprendizado esperada de uma aplicação em que os dados estejam limpos e em amostras suficientes para um bom desempenho, enquanto a linha vermelha (tracejada) indica a curva de aprendizado real obtida a partir de configurações experimentais problemáticas (pela inserção de dados incorretos ou por amostras insuficientes de dados limpos).

Para tratar incorreções de coluna única como erros de digitação, uma diversidade de algoritmos e soluções foram propostas. Exemplos incluem a ferramenta OpenRefine [12], que se utiliza de algoritmos de *clustering* para a verificação de similaridades, agrupando os dados utilizando métodos de vizinho mais próximo para sugerir ao usuário possíveis representações de dados que se tratam da mesma entidade, e o DBoost [3], um algoritmo veloz que detecta *outliers* em um conjunto de dados, facilitando a detecção de possíveis erros.

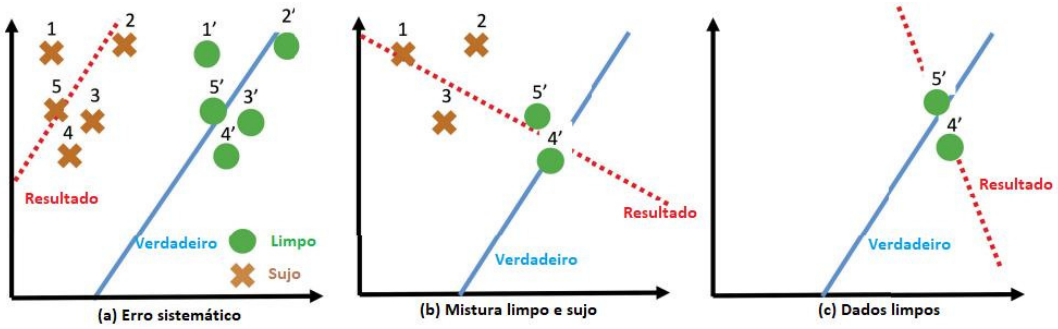

Figura 2 – Deslocamento de modelos de aprendizado de máquina por dados corrompidos. (a) Corrupção sistemática em uma variável é o suficiente para levar a um modelo deslocado. (b) A mistura de dados limpos e sujos resulta em um modelo de menos acurácia que nenhuma limpeza. (c) Amostras insuficientes de apenas dados limpos podem resultar em problemas similares. (Fonte: [2])

2.3.2 Erros de dependência

Algoritmos de detecção de erros em coluna única conseguem tratar problemas como erros de digitação e valores faltantes, mas são desprovidos de detecções de erros estruturais. Como resultado, boa parte dos erros não são detectados.

Dependências são metadados que descrevem associações de determinação de valores entre atributos de uma relação [1]. Exemplos típicos incluem dependências funcionais e dependências de inclusão [1, 13].

Uma dependência funcional é uma relação entre dois atributos dentro de uma mesma relação, de forma que um dos atributos é considerado chave e determina unicamente o valor do outro. Considere um esquema relacional R com atributos $attr(R)$. Matematicamente, uma dependência funcional pode ser definida pela relação

$$X \rightarrow Y \mid X \subseteq attr(R) \text{ e } Y \subseteq attr(R). \tag{2.4}$$

Por exemplo, dado o chassi de um determinado carro, espera-se que este se refira a um fabricante específico. Portanto, o $chassi(X)$ é a chave dessa relação e o atributo $fabricante(Y)$ possui dependência funcional com a coluna *chassi*.

Um erro de dependência funcional ocorre quando há a inserção de múltiplos valores de dependência para um atributo cuja chave já foi estabelecida.

	chassi	**fabricante**	**cor**	**modelo**
1	chassi01	Volkswagen	preto	Jetta 2018
2	chassi02	Ferrari	vermelho	488 GTB 2016
3	chassi03	Hyundai	prata	HB20 2020
4	chassi04	BMW	azul	BMW 320i 2022
5	chassi04	Hyundai	azul	BMW 320i 2022
6	chassi05	Ferrari	azul	BMW 320i 2022

Tabela 1 – Tabela de dados com erros de inserção. (Fonte: Elaboração própria).

Observe a Tabela 1. Suponha que a dependência funcional $chassi \rightarrow fabricante$ vale para o conjunto. Neste caso, a inserção do conjunto de tuplas {“chassi04”, “BMW”, ...}, {“chassi04”, “Hyundai”, ...} é um erro de dependência pois a relação $chassi \rightarrow fabricante$ determina que o valor do atributo fabricante só pode estar atrelado a um único determinado chassi. Note que o contrário não é verdadeiro. Um chassi determina seu fabricante, mas um fabricante não determina o valor do chassi. Por isso, inserções com chassis diferentes e fabricantes iguais são perfeitamente aceitáveis.

Outro tipo de dependência são as dependências de inclusão. Dependências de inclusão são instruções na qual algumas colunas de uma relação estão contidas em outras colunas, tipicamente através de chave estrangeira. Por exemplo, dado uma tabela “peças” e outra tabela “peças-fornecedor”, ambas com os atributos “número-da-peça” referindo-se à mesma entidade, diz-se que essas duas colunas possuem dependências de inclusão entre si, pois a inserção em uma tabela implica na existência e/ou inserção na outra.

Idealmente, restrições de dependência em bases de dados são conhecidas no momento de sua criação, possibilitando ao programador especificá-las, evitando inserções que violem regras de negócio. No entanto, muitas bases de dados não vêm com restrições de dependência explícitas ou conhecidas no momento de sua criação, viabilizando erros. Existem algumas ferramentas como a DC-Clean [14] que oferecem ao usuário a possibilidade de inserção de regras de qualidade e restrições de negação, porém necessitam que o usuário as conheça previamente. Uma alternativa é a pesquisa por dependências feita de forma manual, mas esse processo pode ser bastante custoso. Por isso, várias pesquisas já foram feitas para a detecção automática de dependências em conjuntos de dados [15, 16], incluindo algumas mais recentes com foco em aplicações de *big data* [17, 18]. Existem também alternativas como a KATARA [19], que buscam verificações semânticas utilizando-se de bases externas de conhecimento e contam com validação humana para o reconhecimento de dependências.

2.3.3 Restrições de negação: uma melhoria em representações de dependência

Restrições de dependência funcional e dependência funcional condicional são ótimas para representar a maioria dos tipos de associações entre atributos de uma relação e foram bastante exploradas em estudos passados [20, 21, 16]. No entanto, em um estudo publicado em 2013, Chu et. al [22] demonstram como dependências funcionais são limitadas em poder de expressão e propõem um novo algoritmo de detecção de erros baseado em restrições de negação (*DCs*, do inglês *denial constraints*).

DCs possuem maior capacidade de representação do que restrições de dependências funcionais, sejam elas condicionais ou não. Considere uma tabela de dados com os dados utilizados pelo departamento de Recursos Humanos de uma empresa, representada a seguir.

	cpf	nome	cidade	estado	dpt	funcao	salario	num_vendas	bonus_comissao
1	123456789-1	Wilson Andrade	São Paulo	São Paulo	Computação	Desenvolvedor Jr.	3500	100	1500
2	123456789-2	William Medina	Londrina	Paraná	Computação	Desenvolvedor Jr.	4200	90	1350
3	123456789-3	João Arruda	Petrópolis	Rio de Janeiro	Vendas	Vendedor Sr.	9000	550	22000
4	123456789-4	Henrique Gonçalves	Londrina	São Paulo	Recursos Humanos	Psicólogo	2500	0	0

Tabela 2 – Tabela de dados com informações de funcionários de uma empresa. (Fonte: Elaboração própria).

Suponha que os atributos armazenados são o cpf, nome, cidade, estado, departamento, função, salário, número de vendas e um bônus de comissão. Suponha também que as seguintes restrições de integridade valem para o conjunto:

a) o cpf identifica o nome de uma pessoa;

b) se a cidade de uma pessoa é Londrina, ela reside no estado do Paraná;

c) se dois funcionários trabalham no mesmo departamento e têm a mesma função, um não pode ter salário maior do que o outro;

d) o bônus de comissão de cada funcionário está limitado ao seu teto salarial.

As restrições a) e b) podem ser expressas por restrições de dependência funcional e dependência funcional condicional, respectivamente, pelas seguintes cláusulas:

a) $CPF \rightarrow nome$;

b) $Cidade \rightarrow Estado(Londrina||Paraná)$.

As restrições (c), (d) e (e), no entanto, dependem de predicados de ordem (menor que, menor ou igual a, maior ou igual a e maior que) e, por isso, não podem ser expressas por dependências funcionais ou dependências funcionais condicionais, mas podem ser expressas por lógica de primeira ordem, da seguinte maneira:

c) $\forall t_\alpha, t_\beta \in R, \neg(t_\alpha.departamento = t_\beta.departamento \wedge t_\alpha.função = t_\beta.função \wedge t_\alpha.salário > t_\beta.salário)$;

d) $\forall t_\alpha \in R, \neg(t_\alpha.bonus\_comissao > t_\alpha.salário)$.

Segundo Chu et. al [22], todo e qualquer conjunto de dependências funcionais pode também ser representado por DCs através de transformações sintáticas. Para o caso de (a) e (b), as respectivas transformações seriam:

a) $\forall t_\alpha, t_\beta \in R, \neg(t_\alpha.cpf = t_\beta.cpf \wedge t_\alpha.nome \neq t_\beta.nome)$;

b) $\forall t_\alpha \in R, \neg(t_\alpha.cidade =' Londrina' \wedge t_\alpha.estado \neq' Paraná')$.

Devido a seu maior poder de expressividade, DCs também possuem espaço de procura maior e tornam tarefas de busca por violações em dados mais computacionalmente intensivas [22, 23]. Por isso, algoritmos que buscam a automatização de descobertas de DCs geralmente aplicam algum algoritmo de otimização de busca ou heurística parcial. O algoritmo FastDC [], por exemplo, mesmo sendo a primeira proposta de busca por DCs automatizada, utiliza uma estratégia *depth-first search* para auxiliar no processo de descoberta de violações.

Outra proposta para a detecção automática de conjuntos de DCs é o algoritmo Hydra [24]. O algoritmo Hydra consegue otimizar a complexidade de tempo quadrática necessária para a detecção de DCs presentes no algoritmo do FastDC e, segundo os autores, torna a complexidade de tempo necessário perto de linear. Isso é feito através da redução de comparações entre tuplas que o Hydra considera como redundante. Dois pares de tuplas são considerados redundantes quando eles violam exatamente o mesmo DC candidato; para estes casos, é suficiente que se considere apenas uma das tuplas.

Há ainda uma proposta mais recente no campo de detecções automáticas de DCs: o algoritmo DCFinder [23]. A ideia é a descoberta de restrições de negação parciais, ou seja,

DCs que são apenas parcialmente satisfeitas. Dessa forma, é possível otimizar o algoritmo de busca consideravelmente, pois o espaço de procura é reduzido.

2.4 Aprendizado de máquina

O aprendizado de máquina diz respeito ao método computacional que, dado um conjunto de dados de teste e treinamento, possibilita o aprimoramento de uma dada tarefa computacional [25] através do reconhecimento de padrões. Essa tarefa geralmente consiste em realizar alguma previsão. Por exemplo, suponha que em um programa computacional foram analisadas as características faciais de jovens entre 18 a 25 anos. Dada a tarefa de determinar se um rosto combina com um jovem, um algoritmo de aprendizado de máquina pode – usando as informações coletadas – prever se a pessoa é jovem, mesmo que nunca tenha sido vista, pois suas características esperadas são conhecidas.

2.4.1 Limpeza de dados com aprendizado de máquina

Estudos têm mostrado que não existe uma única melhor aplicação de métodos de limpeza devido às dependências relacionais de dados. Mesmo quando várias ferramentas são aplicadas, o desempenho é frequentemente ruim [4] se não for capturada a natureza holística do processo de limpeza. Por isso, técnicas que se utilizam de aprendizado de máquina têm apresentado uma direção interessante de linha de pesquisas, particularmente para a preparação e limpeza de dados. O ActiveClean [2] é um sistema que propõe uma abordagem baseada em aprendizado de máquina para a correção/reparo de erros em conjuntos de dados. Com um modelo inicial de entrada, uma função de caracterização e um conjunto sujo de dados, o ActiveClean, através de levantamentos estatísticos, determina possíveis células errôneas. O analista então limpa lotes de dados colhidos, que são realimentados no sistema para retreinar de forma inteligente o modelo gerado, via um processo de gradiente descendente estocástico.

Mais recentemente, propostas como o HoloDetect [5] propuseram técnicas de *few-shot learning* (um problema de aprendizado de máquina onde, com um conjunto limitado de exemplos para treino, busca-se aprender soluções gerais) para detecção de erros, com valores de precisão e revocação em torno de 90% para alguns conjuntos sintéticos de dados. Tais propostas também necessitam de ajustes efetuados pelo usuário, como a seleção ou configuração do algoritmo. Para limitar ao máximo a necessidade de interação humana, Mohammad et al. propõem um algoritmo livre de configurações, de forma semissupervisionada, com a utilização das ferramentas Raha [8] e Baran [26]. Primeiro, o Raha gera um conjunto inicial de erros potenciais através da utilização de alguns detectores. Estes detectores são uma lista de ferramentas e algoritmos determinados pela ferramenta Raha, e aumentam particularmente o limite de recuperação alcançável da tarefa de detecção de erros quando comparado a outras abordagens. Em seguida, a saída desses detectores de

erros básicos é agrupada em um conjunto final de erros potenciais e o usuário corrige os erros de uma maneira semissupervisionada. As ferramentas Raha e Baran pedem iterativamente ao usuário para anotar uma tupla, ou seja, marcar e corrigir alguns conjuntos que ele considerou como erro. As ferramentas Raha/Baran então aprendem a generalizar os exemplos de correção de erros fornecidos pelo usuário para o resto do conjunto de dados, sem auxílio do usuário. Por se tratar de uma ferramenta escolhida para o estudo do presente trabalho, mais detalhes sobre o processo de execução da ferramenta Raha podem ser observados na seção 3.1.1.

Mesmo com o avanço de pesquisas na área de aprendizado de máquina, há ainda problemas que se destacam. Por exemplo, sabe-se que algoritmos de aprendizado de máquina necessitam de certa relevância entre dados. No entanto, a natureza flexível das grandes empresas e negócios implica em mudanças constantes em fluxos de informação. Dados que possuem muita maleabilidade constituem barreiras para a criação de algoritmos com base em aprendizado de máquina pois a detecção de itens corretos e incorretos é uma tarefa não trivial. Além do mais, existem conjuntos de dados que exigem conhecimento estrutural da organização dos dados, de forma que usuários leigos não conseguiriam identificar problemas, aumentando a dificuldade de criação de algoritmos de aprendizado de máquina eficientes.

2.5 O problema estrutural de pesquisas em limpeza de dados

Propostas de ferramentas e algoritmos de detecção de erros em bases de dados na literatura mais recente utilizam conjuntos diferentes de dados sob configurações experimentais distintas [5, 8, 9] . No entanto, dependendo do tipo e quantidade de erros introduzidos, variações na capacidade de detecção de erros de cada ferramenta podem ser significativas [4]. Na publicação da ferramenta Raha [8], por exemplo, a ferramenta DBoost obteve medida F1 de 0,49 quando submetida ao conjunto de dados Hospital, mas apenas 0,16 quando testado no conjunto de dados Rayyan. Outros estudos utilizam conjuntos de dados diferentes e demonstram variações que tornam difícil ao analista de dados distinguir qual ferramenta será melhor para a sua aplicação.

Outro fator que pode alterar significativamente a capacidade de detecção de uma ferramenta é o tipo de erro introduzido. Estudos anteriores [8, 5, 4] não deixam claro o impacto que a mudanças específicas causam ao desempenho de ferramentas de detecção de erros, pois esses alteram não somente um conjuto específico e controlado de valores, mas conjuntos de dados inteiros, modificando simultaneamente vários fatores como cardinalidade, tipos de dado, quantidade de erros e até regras de negócio entre uma execução e outra. Apesar de abordagens como essa fornecerem uma ideia geral da capacidade de detecção de cada ferramenta para erros variados, elas falham em determinar o impacto que a porcentagem e distribuição de cada erro pode causar ao desempenho de cada detector.

3 PROPOSTA DE SOLUÇÃO UNIFICADA E METODOLOGIAS DE AVALIAÇÃO

Devido à quantidade reduzida de estudos englobando ferramentas do estado da arte sob um único ambiente de testes e a necessidade de estudos que demonstrem impactos de alterações específicas como a quantidade ou tipo de determinado erro em conjuntos de dados relacionais, faz-se necessário a criação de um conjunto de experimentos que possam extrair informações que demonstrem o impacto dessas alterações ao desempenho de detecção de erros de cada ferramenta, bem como ofereçam um comparativo direto entre o desempenho/comportamento de cada uma, proporcionando uma visão mais ampla ao analista de dados sobre o que esperar de cada sistema de detecção de erros. Os detalhes do experimento desenvolvido são descritos a seguir.

3.1 Ferramentas

Existem diversas ferramentas apresentadas no mercado e na literatura. Devido ao escopo do projeto, fez-se necessário a filtragem para um conjunto seleto de sistemas. Os critérios para escolha das ferramentas utilizadas seguiram a seguinte ordem, com preferência de cima para baixo e com todas as condições não excludentes:

a) a ferramenta está disponível e é de acesso livre ou possui versão experimental;

b) a ferramenta é a mais recente possível;

c) a ferramenta possui meios de comparação direta com outras ferramentas já selecionadas.

Durante o levantamento bibliográfico para este estudo, notou-se que a grande maioria das abordagens apresentadas recentemente, como o AutoDetect [27] e o HoloDetect [5], não possuem código fonte disponível ou estão disponíveis apenas comercialmente. Por isso, nem todas as ferramentas que foram utilizadas para este estudo são abordagens novas ou que se utilizam de aprendizado de máquina. As ferramentas finais escolhidas são descritas a seguir.

3.1.1 Raha

O Raha [6] é uma ferramenta do ano de 2019, sendo a mais recente das selecionadas. Ela toma como parâmetros um conjunto de dados sujo S e um número N chamado de orçamento de rotulagem, cujo valor representa o número de tuplas com as quais o usuário interagirá. O algoritmo baseia-se em três etapas fundamentais: execução automá-

tica de algoritmos, amostragem de tuplas e correções pelo usuário junto a propagação de resultados.

A primeira etapa consiste na execução automática de algoritmos. A ferramenta Raha roda um conjunto S de estratégias de detecção de erros. Segundo os autores, Raha é capaz de sistematicamente detectar e configurar algoritmos existentes de acordo com o tamanho e tipo de conjunto de dados. Desta forma, é possível otimizar o tempo de execução do algoritmo, filtrando-se os parâmetros de execução e decisão de acordo com o necessário. Dentre as possíveis estratégias de detecção listam-se algoritmos de detecção de *outliers*, violações de padrões, violações de regras e detecção de violações baseadas em bases de conhecimento.

Cada estratégia $s \in S$, ou marca uma célula $c[i, j]$ como um erro ou não. Formalmente, essa informação é gravada como

$$s(c[i,j]) = \begin{cases} 1, & \text{sse } s \text{ marca } c[i,j] \text{ como erro;} \\ 0, & \text{caso contrário.} \end{cases}$$

O Raha então guarda os resultados de cada estratégia em conjuntos de *feature vectors* (vetores de característica). O vetor de características da célula c[i, j] é o vetor com todas as saídas de estratégias de detecção $s \in S$ nessa célula. Formalmente,

$$(c[i,j]) = [s(c[i,j]) \mid \forall s \in S]. \tag{3.1}$$

Portanto, o conjunto de vetores de característica de células $(d[i, j])$ dentro de uma coluna de dados em particular j é

$$Vj = \{v(c[i,j]) \mid 1 \leq i \leq |c|\}. \tag{3.2}$$

A segunda etapa consiste na amostragem de tuplas a apresentar ao usuário, limitando-se a N tuplas estabelecidas. Para cada tupla amostrada, o usuário corrige todas as células da tupla. O algoritmo determina quais tuplas mostrar ao usuário baseando-se em agrupamentos de células similares, provenientes da etapa anterior. Quanto menos similares os agrupamentos forem, maior a probabilidade de se tratarem de tipos de erros diferentes. Dessa forma, é possível evitar ao máximo a repetição de erros, aproveitando melhor correções do usuário.

Por fim, a terceira e última etapa consiste na propagação das correções feitas pelo usuário ao conjunto de agrupamentos. Um treino utilizando técnicas de aprendizado de máquina é executado e o algoritmo prediz os valores do restante das células dos conjuntos, gerando um conjunto resultante R.

3.1.2 Trifacta Data Wrangler

O Trifacta Data Wrangler [28] é uma ferramenta visual desenvolvida para a transformação de dados para aplicações de limpeza de dados. Três versões são oferecidas: o Trifacta Wrangler, o Trifacta Wrangler Pro, e o Trifacta Wrangler Enterprise. O Wrangler Pro e o Wrangler Enterprise são versões comerciais. Por isso, a versão que foi utilizada foi a versão básica do Trifacta Wrangler, com a utilização de uma conta acadêmica.

A ferramenta Trifacta Data Wrangler consegue predizer, sugerir e aplicar transformações sintáticas em conjuntos de dados fornecidos pelo usuário. A Figura 3 demonstra a página inicial da ferramenta após o usuário inserir um conjunto de dados sujo e clicar em uma coluna que foi identificada como possuidora de erros.

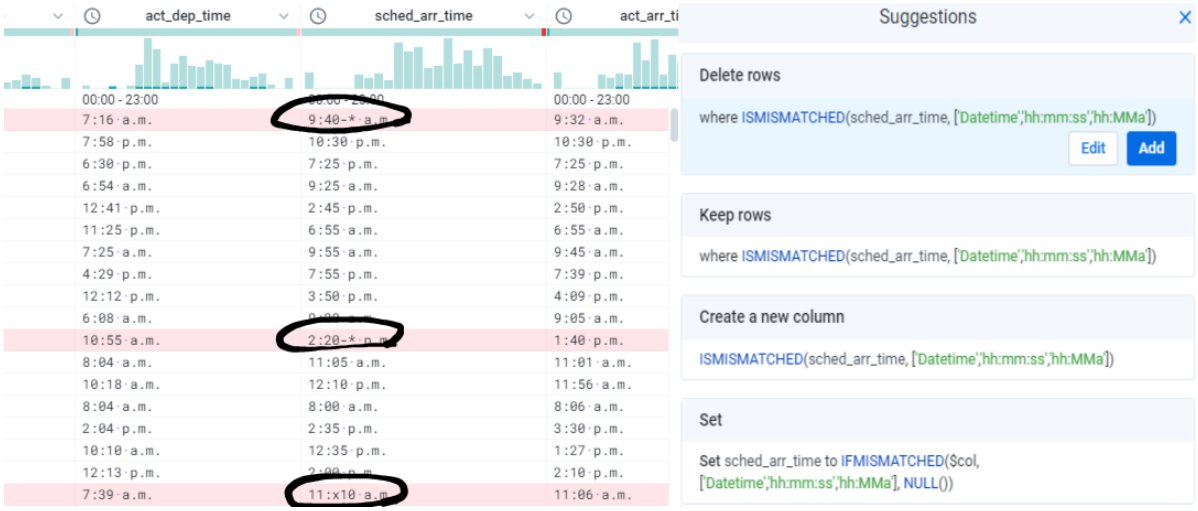

Figura 3 – Ferramenta Trifacta identifica células fora dos padrões esperados na coluna "sched_arr_time" e sugere alterações. (Fonte: Elaboração própria).

Após uma análise automática de frequência de valores e a execução de um mapeamento de padrões de expressões regulares (coluna a coluna), a ferramenta fornece sugestões ao usuário com informações de células que ela julga como possivelmente incorretas. O usuário pode filtrar esses valores e analisá-los um a um ou manter o conjunto de dados completo.

Outra funcionalidade da Trifacta Data Wrangler são "funções de padronização". Ao clicar com o botão direito sob uma coluna e navegar até a opção "*standardize*", a ferramenta roda um algoritmo de *ngram-fingerprint* [29] sobre cada atributo da coluna - uma função de agrupamento por similaridade - e gera um grupo de *clusters* de similaridade os quais ela apresenta ao usuário. O usuário pode então alterar manualmente o valor de cada agrupamento encontrado ou rodar uma "padronização automática", onde cada valor de agrupamento é decidido automaticamente via um processo estatístico. Se a diferença de contagem entre um atributo e sua contraparte for suficientemente grande, o atributo

com menor ocorrência de valores é substituído por aquele com maior ocorrências dentro do seu *cluster*. Um exemplo desse processo é demonstrado na Figura 4.

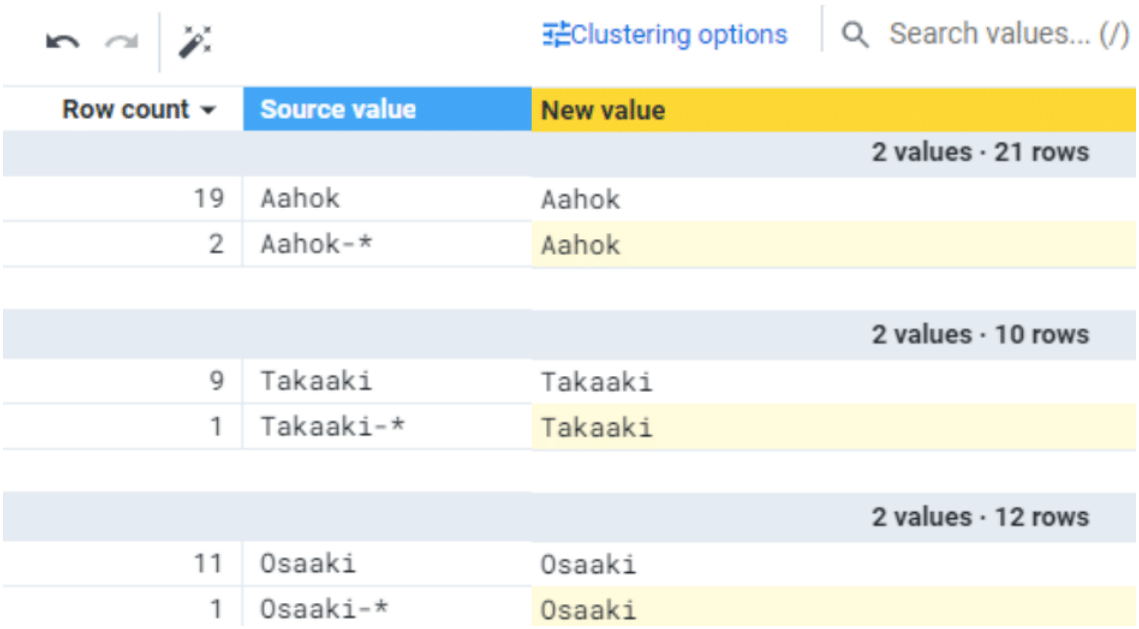

Figura 4 – Resultado de agrupamento por padronização automática. (Fonte: Elaboração própria).

Por fim, o Trifacta Data Wrangler consegue identificar valores que estão fora do tipo de atributo esperado. Em uma coluna com noventa e cinco valores do tipo *string* e cinco valores numéricos, por exemplo, o Trifacta Data Wrangler acusará os cinco atributos numéricos como "fora do padrão esperado", pois a diferença de frequência entre o tipo de dado encontrado e outros tipos presentes na coluna é suficientemente grande.

3.1.3 ED2

O ED2 [9] é o resultado de um caso de estudo feito por pesquisadores da Universidade Técnica de Berlin em conjunto com estudantes da *German Research Centre for Artificial Intelligence (DFKI)*, e é uma das ferramentas que se utilizam de técnicas de aprendizado de máquina, nomeadamente aprendizado ativo, para a detecção de erros em conjuntos de dados relacionais.

O ED2 trabalha com uma análise coluna a coluna e a aprendizagem é feita através de um processo dividido em dois estágios: primeiramente um componente seletor de colunas escolhe uma coluna promissora para aprendizado e depois um segundo refinamento estatístico é feito por um gerador de lotes para determinar quais conjuntos de células são mais promissoras para correções do usuário. O processo do funcionamento do algoritmo ED2 é explicado a seguir.

Primeiro, similarmente ao Raha, um extrator de características gera vetores de características de cada célula. Enquanto o Raha gera vetores através de resultados de algoritmos pré-selecionados, o ED2 extrai características a partir de modelos n-grama e metadados sobre o conjunto de dados de entrada. Formalmente, o vetor de características gerado pelo modelo n-grama é definido pela fórmula

$$\rho_n - gramas(c[i,j],n) = [tf(\omega, c[i,j]) \cdot idf(\omega, c[:,j]) \mid \omega \in \Omega_n(c[:,j])], \tag{3.3}$$

onde $\Omega_n(c[:,j])$ é o conjunto de todos os n-gramas ω de tamanho n que existem na coluna $c[:,j]$. A frequência de termos tf mede o número de vezes que um n-grama ω específico ocorre na célula $c[i,j]$ e a frequência inversa de termos idf mede se um n-grama ω é comum ou raro nas células da coluna $c[:,j]$.

Quanto a metadados, as características obtidas são: contagem de ocorrências, o tipo de valor,o tamanho da *string* se o dado for do tipo textual, e o valor numérico de representação da célula.

Com o vetor de características e a arrecadação de metadados, o algoritmo do ED2 determina um conjunto inicial de células para apresentação ao usuário. Em contraste à ferramenta Raha, o usuário não precisa fornecer o resultado da célula. O usuário está restrito a informar se a célula está correta ou não. Depois, o ED2 treina um classificador por coluna e otimiza seus modelos de classificação de acordo com as correções feitas pelo usuário. O algoritmo então estima a probabilidade de células da coluna analisada estarem incorretas e somente então dá início ao processo de aprendizado ativo.

Um seletor de colunas, baseando-se em um método de seleção que o programador escolhe, define a próxima coluna a ser analisada no processo de aprendizagem ativa. O seletor possui cinco técnicas de busca possíveis, descritas a seguir.

a) Certeza Mínima - o modelo de classificação retorna um “nível de certeza” para cada célula de uma coluna, e o seletor de colunas escolhe a coluna com menor nível de certeza final;

b) Mudança de Previsão Máxima - o seletor de colunas escolhe a coluna com o maior número de mudanças de previsão. Mudança de previsão é a fração de predições que mudaram comparadas à iteração anterior de aprendizado ativo;

c) Escolha Aleatória - o seletor escolhe a próxima coluna aleatoriamente;

d) Round-Robin - o seletor seleciona a próxima coluna de forma a manter cada coluna com quantidades equilibradas de escolhas. Esse método evita que colunas sejam selecionadas desproporcionalmente;

e) Erro Máximo - a ferramenta executa validações cruzadas entre células de cada coluna e determina uma medida F1 resultante. A coluna escolhida é a que possuir menor medida F1 proveniente das validações.

Após a definição da coluna a ser utilizada para aprendizagem ativa, um gerador em lotes seleciona células promissoras para análise do usuário. A técnica utilizada pelo gerador é um algoritmo *query by committee* [30]: uma variedade de modelos são treinados nos dados atualmente rotulados, e votam um valor sugerido para a saída em dados não rotulados. As células finais escolhidas para treinamento pelo usuário são aquelas em que os modelos mais divergem em suas sugestões, e portanto, possuem menor grau de certeza.

O processo de aprendizagem ativa é repetido enquanto o usuário fornecer novas correções, ou até que as células para aprendizagem ativa se esgotem.

3.1.4 HoloClean

O Holoclean [11], em contraste às outras ferramentas, não é um detector de erros, mas um *framework* para a correção e reparo de dados inconsistentes. A ferramenta toma como entrada um conjunto de dados sujo e um arquivo de entrada contendo uma lista de DCs, que são utilizados pelo sistema para a detecção de erros.

A versão original do sistema constrói um Modelo Gráfico Probabilístico (GPM) [31] cujas variáveis aleatórias capturam a incerteza sobre as células no conjunto de dados de entrada. Especificamente, dado um conjunto de dados D, o sistema associa cada célula $c \in D$ a uma variável aleatória Tc que recebe valores de um domínio finito $dom(c)$ e compila um PGM que descreve a distribuição de variáveis aleatórias Tc.

A versão corrente do HoloClean utiliza um modelo de ML semelhante a uma rede neural de uma camada, que aplica uma função exponencial envolvendo os *features* que descrevem a célula de dados a ser reparada e os pesos aprendidos pelo modelo, seguida de uma função softmax, que produz a probabilidade de cada valor candidato a ser o valor correto para a célula.

3.1.5 DBoost

O DBoost [3] é um sistema para detecção de valores *outliers* baseado em inferência e modelagem estatística de conjuntos de dados. Bancos de dados tipicamente possuem diversos conjuntos de valores de tipos diferentes e, segundo os autores, o fato de uma grande gama de tipos de dados serem passíveis de representação via tipos básicos em linguagens SQL (uma *string*, por exemplo, pode representar desde datas até locais de uma cidade) dificulta o processo de detecção de erros para algoritmos baseados em meios estatísticos, pois poucas informações contextuais podem ser reconstruídas. O DBoost tenta solucionar esse problema através da expansão semântica de tipos básicos SQL para

conseguir informações mais ricas sobre o conjunto de dados. Com isso, a capacidade de detecção de erros do algoritmo aumenta, uma vez que esse processo torna possível a recuperação de uma série de metadados referentes ao conjunto. Dado uma coluna de dados que são datas mas estão representadas em formato numérico, por exemplo, onde a maioria dessas são referentes ao ano de 2022, é possível, após um processo de expansão semântica, identificar esses valores como sendo do tipo data e identificar que valores que divergem muito do esperado provavelmente são erros.

Após uma expansão semântica, o algoritmo executa uma análise estatística de cada coluna da tabela de dados sujos fornecida pelo usuário e estima se existem colunas com alto nível de correlação. Esse processo também consegue detectar possíveis violações de dependência entre dados de diferentes colunas, pois se duas colunas são altamente correlatas, o conjunto de valores dessas colunas provavelmente são determinados um pelo outro. Depois, o algoritmo treina, com a ajuda de estatísticas e metadados retirados dos passos anteriores, um dentre três possíveis modelos fornecidos como parâmetro pelo usuário: distribuição gaussiana, histograma, ou distribuição gaussiana multivariada. Cada modelo é explicado a seguir.

a) distribuição gaussiana: o usuário fornece um valor numérico de desvio padrão e o DBoost supõe que cada célula de dados foi extraída de uma distribuição normal com uma determinada média e desvio padrão. Valores acima do desvio padrão da distribuição normal são considerados como corretos, e valores abaixo do desvio padrão fornecido são considerados como *outliers*;

b) histograma: o usuário fornece uma proporção de pico p e uma proporção de consideração s. O DBoost executa uma contagem de ocorrências de cada valor único em cada coluna da tupla expandida e para cada agrupamento de subtuplas possivelmente correlacionadas, sem fazer suposições sobre os dados em estudo [3]. Depois, o DBoost filtra os histogramas em um subconjunto de "histogramas úteis", que são decididos através da aplicação de uma heurística própria e o auxílio da proporção de pico p fornecida pelo usuário. Para cada histograma pertencente aos histogramas filtrados, os valores que encontram-se abaixo da proporção de consideração s fornecida pelo usuário são considerados como *outliers*;

c) mistura gaussiana multivariada: o usuário fornece um valor ns de suboperações gaussianas e um limite l, e o DBoost executa um modelo de mistura gaussiana multivariada, reportando valores cuja probabilidade está abaixo do limite l, conforme predito por cada modelo de dados formados por ns suboperações gaussianas.

A Figura 5 mostra uma visualização simplificada de cada modelo para efeitos de ilustratação.

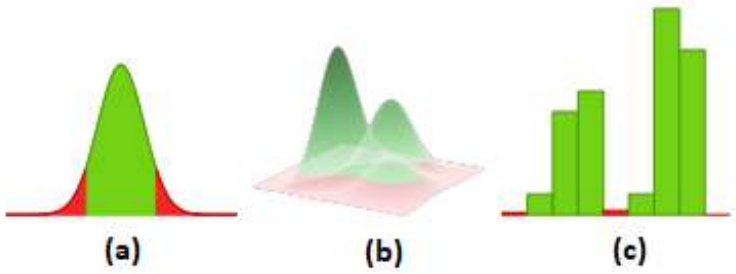

Figura 5 – Visualização simplificada de cada modelo de detecção de *outliers* utilizado pelo DBoost. (a) Modelo de distribuição gaussiana. (b) Modelo de mistura gaussiana multivariada. (c) Modelo de histograma. Pigmentações verdes indicam valores aceitos como corretos. Pigmentações vermelhas indicam região de valores detectados como *outliers*. (Fonte: [3]).

3.2 Conjuntos de dados

Esta seção descreve os conjuntos de dados utilizados para os experimentos executados. O critério de escolha para os conjuntos de dados foi a relevância do conjunto em estudos anteriores, variação na quantidade de células, cardinalidade de colunas, tipos de dados e relações de dependência entre colunas. Também foi estabelecida a utilização de pelo menos um conjunto com dados numéricos, para análise do comportamento de cada ferramenta com dados numéricos errôneos. Os conjuntos de dados utilizados, bem como suas restrições de dependência, estão disponíveis[1] e são descritos a seguir.

Flights é um conjunto de dados contendo informações sobre a hora de partida e chegada de voos de diferentes fontes de dados reais. Foi utilizado em estudos como Raha [8] e HoloClean [11]. Esse conjunto é composto por 2376 tuplas e 6 colunas textuais. Além disso, erros de dependência funcional foram inseridos de forma equilibrada, violando as seguintes 4 restrições de integridade: um mesmo voo não pode ter horários de partida estimada conflitantes; um mesmo voo não pode ter horários de partida reais conflitantes; um mesmo voo não pode ter horários de chegada estimada conflitantes, e um mesmo voo não pode ter horários de chegada reais conflitantes.

Hospital é um conjunto de dados sintéticos de referência usado em vários trabalhos de limpeza de dados [14][11][5] e é composto por 1000 linhas e 18 colunas. O conjunto possui várias células duplicadas, o que pode teoricamente facilitar o aprendizado de padrões e a descoberta de dependências entre células - em especial para aplicações de aprendizado de máquina. Para este conjunto, foram inseridos erros de dependência de forma equilibrada e violando 14 restrições de integridade: tuplas que referenciam o mesmo hospital não podem ter código de cep, cidade, telefone, número de provedor, endereço ou dono

[1] <https://github.com/williamgrmedina/TCC-William-Medina>

com valores conflitantes; um mesmo código de medida não pode ter nomes de medida, condição ou atributos "stateavg" diferentes; o número de provedor está limitado a um único hospital; cada cidade tem um único nome de município; o código de cep determina o serviço de emergência disponível; o nome de medida determina o código de medida, e duas tuplas com a mesma condição e com mesmo nome de medida referem-se ao mesmo tipo de hospital.

Tax_200k é um grande conjunto de dados sintéticos retirados do repositório BART[2] (Benchmarking Algorithms for (data) Repairing and (data) Translation. Ele foi utilizado no estudo de desempenho do Raha [8]. Tax possui diversas informações esparsas (geradas semi-aleatoriamente de forma sintética) que simulam dados da população dos Estados Unidos da América. O conjunto contém uma mesclagem equilibrada de informações textuais e numéricas, onde 8 colunas são textuais e 7 são numéricas (estas ainda divididas entre números inteiros e números reais). Para este conjunto foram utilizadas 7 restrições de integridade: se o código de área é igual, então o telefone não pode se repetir; códigos de cep iguais pertencem à mesma cidade; códigos de cep iguais pertencem ao mesmo estado; o código de área determina o estado; se dois indivíduos possuem criança(s) e moram no mesmo estado, a taxa de isenção para seus dependentes não pode ser diferente, e uma pessoa que possui salário maior que outra e mora no mesmo estado não deve possuir imposto de renda menor.

Para analisar se o tamanho do conjunto de dados pode afetar a capacidade detecção de uma ferramenta, um subconjunto contendo apenas as primeiras 20.000 tuplas do Tax_200k foi extraído e testado sob as mesmas configurações. Os resultados obtidos se encontram junto aos outros testes executados, no capítulo 4.

3.3 Inserção de Erros

Cada conjunto de dados recebeu a introdução de erros de forma controlada. Os erros foram gerados de forma automatizada pela ferramenta BART [1] via uma série de scripts. O BART [32] é uma ferramenta que, dado um arquivo com instruções de inserção de erros (script), gera erros sintéticos de forma controlada. O usuário estabelece o conjunto de dados a ser utilizado, uma série de regras de integridade (sejam elas DCs ou dependências funcionais simples) referentes a este conjunto, e um conjunto de estratégias S de inserção de erros. Para cada estratégia $s \in S$, o usuário estabelece quais colunas terão erros inseridos e a quantidade de erros que cada coluna receberá.

O presente trabalho considerou todas as possíveis combinações de inserção de erros disponíveis pelo BART e inseriu erros utilizando-se das técnicas de inserção aleatória e de inserção de erros estruturais.

[2] <http://www.db.unibas.it/projects/bart/>

Para cada script de configurações, foi estabelecido que as porcentagens de erros inseridos deveriam variar de 0,5% a 10% (identificando o efeito de variações percentuais na capacidade de detecção de cada ferramenta), e que todas as colunas deveriam participar do processo em igual peso e quantidade. Assim, se em um conjunto foram inseridos erros na taxa de 10%, e este conjunto possui 10 colunas, então cada coluna deste recebeu aproximadamente 1% de erros.

Se um script utilizou uma combinação de estratégias diferentes, então cada estratégia recebeu uma divisão dos erros gerados, em igual peso. Por exemplo, se erros foram gerados na taxa de 10% e duas estratégias foram utilizadas, então cada estratégia inseriu 5% de erros.

As estratégias finais escolhidas para a inserção de erros foram compostas por uma combinação de estratégias de inserção aleatória e de inserção de erros de DCs. Para cada conjunto de dados, foram utilizados os conjuntos de DCs listados na seção 3.2. O algoritmo 1, abaixo, traz uma visão simplificada do processo utilizado para a inserção de erros.

Algoritmo 1: Processo de inserção de erros

```
1  início
2      porcentagens ← [0.5%, 1%, 5%, 10%];
3      Estabeleça as colunas a serem utilizadas para inserção de erros;
       para cada p ∈ porcentagens faça
4          carregue o conjunto de dados limpo;
5          insira p erros utilizando a estratégia "erros aleatórios";
6          salve o conjunto resultante;
7      fim
8      Estabeleça as colunas a serem utilizadas para inserção de erros;
       para cada p ∈ porcentagens faça
9          carregue o conjunto de dados limpo;
10         insira p erros utilizando a estratégia "erros de DCs";
11         salve o conjunto resultante;
12     fim
13     Estabeleça as colunas a serem utilizadas para inserção de erros;
       para cada p ∈ porcentagens faça
14         carregue o conjunto de dados limpo;
15         insira p erros utilizando as estratégias "erros aleatórios" e "erros de
             DCs";
16         salve o conjunto resultante;
17     fim
18 fim
```

3.4 Parametrização de ferramentas e outras configurações

Algumas ferramentas como Raha necessitam de parâmetros complementares à entrada de dados. No caso da ferramenta Raha a decisão de escolha para o orçamento de rotulagem N foi de 50. Este parâmetro serviu também de base para parametrizações em outras ferramentas. Detalhes por trás desta decisão encontram-se no apêndice A.

Para ferramentas não determinísticas, cada teste foi executado 5 vezes. Se o coeficiente de variação na medida F1 resultante das 5 execuções foi maior do que 30%, as execuções foram incrementadas de 5 em 5 até que o nível de variância entre resultados fosse diminuído ao nível estabelecido, ou a um total de 100 execuções. Esse incremento foi especialmente útil para casos de dados com porcentagens de erros muito baixas; segundo observações feitas, quando há porcentagens muito baixas de erros, as vezes nenhum erro é detectado - principalmente com algoritmos de aprendizado de máquina. para estes casos, a medida F1 entre uma execução e outra pode variar significativamente.

Outra ferramenta que se utiliza de dados iniciais além do conjunto de dados é o ED2 [9]. Para esta ferramenta, foram utilizados 25 rótulos iniciais e 25 rótulos de usuário via *Active Learning* (AL), totalizando 50 execuções por um processo similar ao Raha; para o seletor de colunas, foi utilizada a técnica de certeza mínima. Por se tratar de uma ferramenta não determinística, a ferramenta também esteve sujeita a várias execuções.

A ferramenta Trifacta Wrangler [28] é uma ferramenta determinística e, por isso, não necessita de execuções extras. Utilizando detecções de padrões e a função de padronização automática da ferramenta (Figura 3, Figura 4), foi possível indentificar células acusadas como erro pela ferramenta. Em alguns casos, a padronização automática da ferramenta pode não ter certeza ou corrigir células erroneamente, conforme demonstra a Figura 6. Por isso, foi fornecido um "saldo de correção" de 50 correções ao usuário como forma de comparação direta entre outras alternativas de detecção de erros. O "saldo de correção" é a oportunidade dada ao usuário de corrigir manualmente um conjunto de células que a função de padronização automática não conseguiu solucionar ou corrigiu erroneamente gerando falsos positivos. Assim, o usuário foi incluído como entidade participante no processo de decisão de erros, de maneira similar a ferramentas baseadas em métodos de aprendizado de máquina.

O HoloClean [11] - além do conjunto de dados D - depende de um conjunto de DCs referente a D, o estabelecimento de quais detectores de erros DE a ferramenta utilizará, e uma série de parâmetros e especificações referentes ao domínio de células a ser utilizado na pesquisa por violações. Inicialmente, os parâmetros utilizados foram os padrões dados no exemplo de reparo fornecido pelos desenvolvedores do HoloClean na página do projeto[3] da ferramenta. Dois detectores de erros foram utilizados: um para erros de dependência e

[3] <https://github.com/HoloClean/holoclean/blob/master/examples/holoclean_repair_example.py>

Row count	Source value	New value
		2 values · 45 rows
32	APACHE · JUNCTION-*	APACHE · JUNCTION-*
13	APACHE · JUNCTION	APACHE · JUNCTION-*
		2 values · 25 rows
13	Yuugi	Yuugi
12	Yuugi-*	Yuugi-*

Figura 6 – A ferramenta Trifacta corrige erros falsamente ou obtém grau de certeza insuficiente para executar alterações automáticas. Valores originais e suas respectivas contagens estão listados à esquerda; à direita, são listadas as correcções executadas para cada valor.

outro para campos nulos. Já para o conjunto de DCs, este foi variado para cada conjunto. Os DCs foram traduzidos diretamente a partir das violações inseridas nos testes.

Ao longo dos testes, percebeu-se que a memória da máquina estava sendo esgotada para conjuntos de dados maiores, e constatou-se que o motivo era o alto valor no parâmetro "max_domain". Segundo os autores do HoloClean, uma das etapas do processo de compilação é a fase de *grounding*, que enumera todas as possíveis interações entre variáveis aleatórias correlacionadas e as materializa em um GPM[31] que representa a distribuição contínua sobre as variáveis envolvidas. Essa etapa é consideravelmente cara e não escala para conjuntos de dados maiores [11]. Por isso, o HoloClean oferece o parâmetro "max_domain" como meio de podar o número de comparações. Mais detalhes sobre o parâmetro que foi utilizado em cada teste são descritos no capítulo 4.

Para a ferramenta DBoost, foram executados testes com os modelos de gaussiana simples e histograma. Cada modelo foi executado com os parâmetros recomendados pelo próprio sistema em seu exemplo fornecido na página do projeto no github [4]. O capítulo 4 relata os resultados obtidos pela técnica que melhor se adequou a cada conjunto de dados testado.

Todos os testes foram executados em uma máquina de 32GB de RAM, com um processador Intel Core i7, rodando no sistema operacional Ubuntu 18.04.

[4] <https://github.com/cpitclaudel/dBoost>

4 COLETA E DISCUSSÃO DE RESULTADOS

Este capítulo apresenta os resultados obtidos para cada conjunto de testes e discute os resultados coletados.

4.1 Coleta de resultados

As figuras 7, 8, 9 e 10 apresentam os resultados obtidos pela execução de cada algoritmo testado. Para os conjunto de dados Tax_20k e Tax_200k, o algoritmo HoloClean sobrecarregou a memória RAM disponível no computador. Por isso, foi feita a diminuição gradativa do parâmetro max_domain até que fosse possível finalizar todos os testes com algum valor estabelecido. Os valores finais utilizados foram 5 e 1 para os conjuntos tax_20k e tax_200k, respectivamente.

4.2 Discussão

Existem muitos estudos e propostas novas para a solução de erros em bancos de dados, boa parte destas baseadas em aprendizado de máquina. A grande variedade de conjuntos de dados disponíveis e o impacto que os tipos de erros presentes em cada conjunto podem causar a tarefas de detecção, no entanto, favorecem a escolha de conjuntos de dados propícios, de forma que cada ferramenta possa ser exaltada por seus respectivos autores.

Um dos algoritmos que apresenta resultados excelentes em seus estudos, por exemplo, é o Raha. Os estudos obtidos pelos seus autores indicam que a solução adotada por eles possui superioridade expressiva em relação a outras soluções existentes, bastando apenas 20 tuplas rotuladas.

> *"We compare Raha with 4 standalone error detection algorithms and 3 error detection aggregators. Using only 20 labeled tuples, Raha outperforms any other existing solution." (MAHDAVI et. al, 2019, p. 3)*

Estudos feitos pelo presente trabalho, no entanto, apresentam resultados diferentes e demonstram que nem sempre este é o caso. Para conjuntos de dados com porcentagens baixas de erros, por instância, a ferramenta parece não obter resultados tão significativos. Cabe observar que o número de tuplas rotuladas para cada teste no presente estudo também foram maiores, na ordem de 150%.

Outro problema observado - neste caso especificamente para o Raha - foi que conjuntos de dados com baixas porcentagens de erros podem causar variações significativas

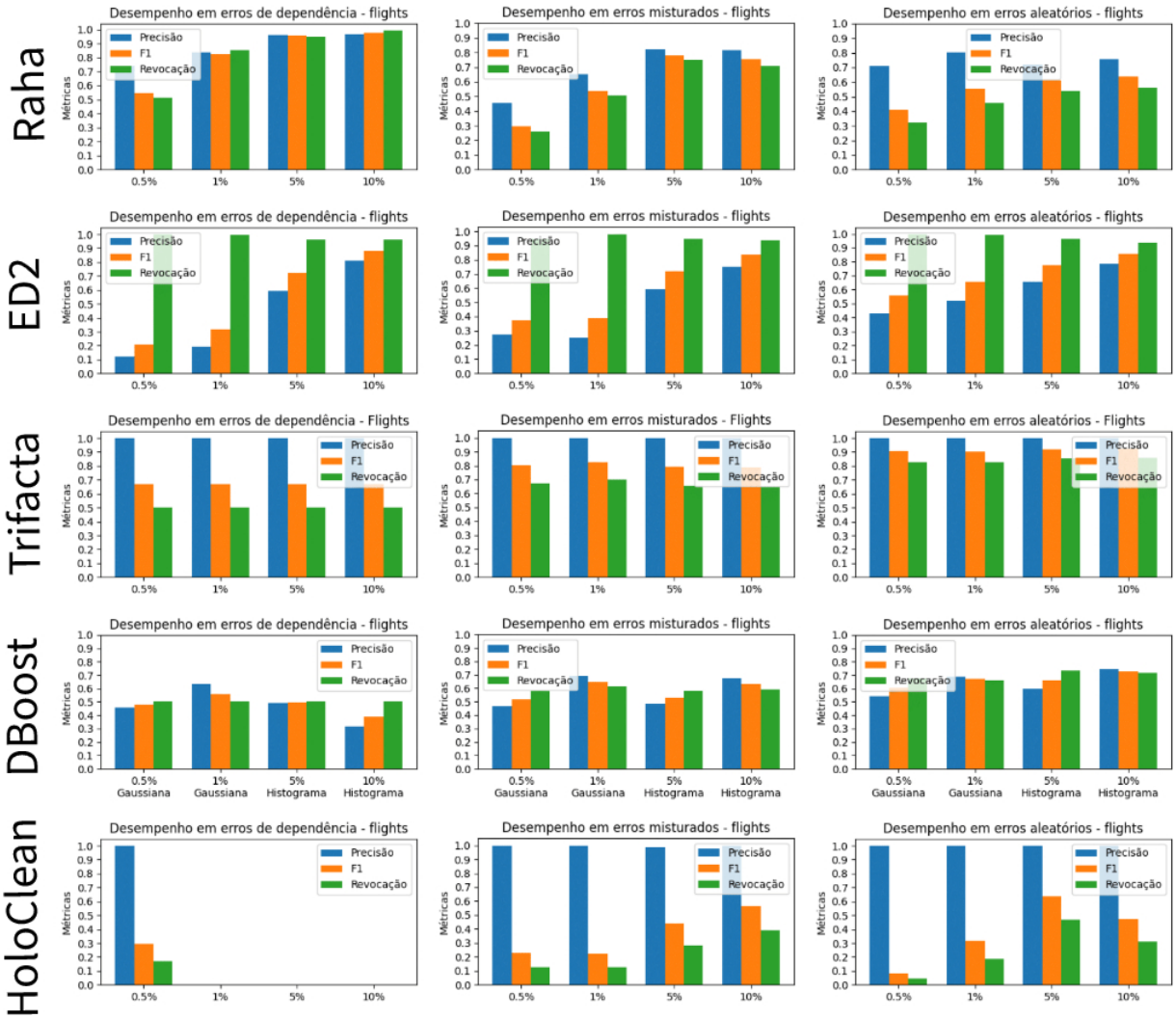

Figura 7 – Resultados obtidos para o conjunto de dados Flights. Feramentas (de cima para baixo): Raha, ED2, Trifacta Data Wrangler, DBoost, HoloClean. (Fonte: Elaboração própria).

na medida F1 entre uma execução e outra, pois a escolha de tuplas a serem rotuladas não é feita de forma determinística. Este fato é exposto no apêndice A. Quanto menor a porcentagem de erros, maior é a variação causada, o que pode ser particularmente frustrante para o cientista de dados que está buscando por erros, uma vez que a reexecução do algoritmo implica na necessidade de uma nova onda de rotulagens.

Outro estudo, feito pelos desenvolvedores da ferramenta ED2, diz que para conjuntos de dados maiores, o ED2 requer uma fração menor de rotulações do que seu concorrente HoloDetect. Não foi possível incluir a ferramenta HoloDetect no presente estudo pois ela não encontra-se disponível ao público geral. Apesar disso, a comparação direta entre o ED2 e o Raha feita no presente estudo demonstrou que esta parece ser uma afirmativa que se confirma, uma vez que o conjunto de dados Tax_200k apresentou medida F1 melhor do que o conjunto Tax_20k para esta ferramenta.

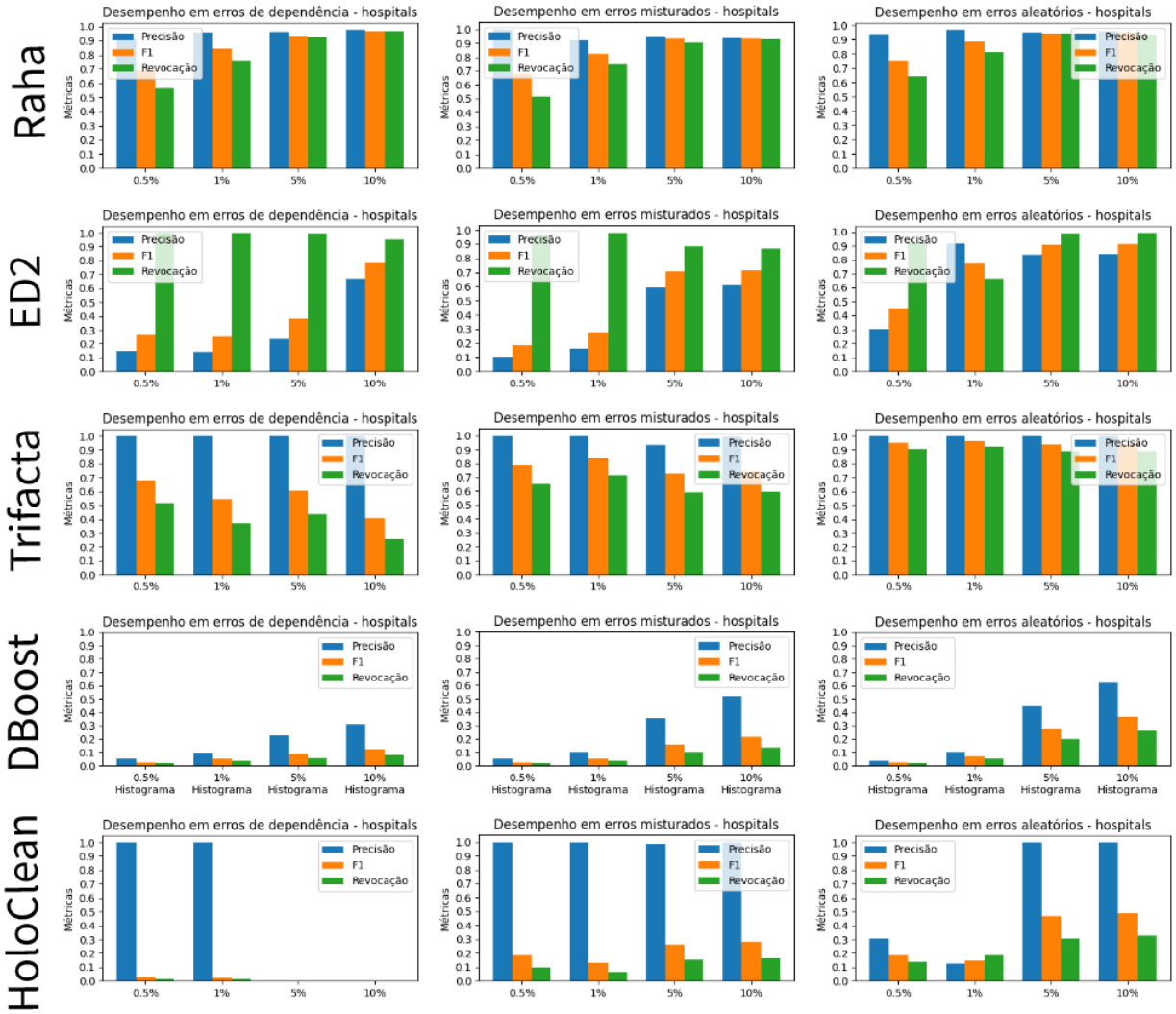

Figura 8 – Resultados obtidos para o conjunto de dados Hospital. Feramentas (de cima para baixo): Raha, ED2, Trifacta Data Wrangler, DBoost, HoloClean. (Fonte: Elaboração própria).

Nenhum estudo anterior executou comparações de desempenho entre o Raha e o ED2. Neste sentido, os resultados obtidos pelo presente trabalho apresentam algumas considerações e novidades importantes.

a) o Raha e o ED2 não diferem significativamente em termos de desempenho. Apesar disso, para conjuntos menores, o Raha apresenta resultados superiores de detecção;

b) apesar da capacidade de detecção de erros do Raha ser superior ou similar ao da ferramenta ED2, o ED2 parece se beneficiar de uma variação muito menor na medida F1 (em relação ao Raha) quando a porcentagem de erros é baixa;

c) quando atrelada a heurísticas probabilísticas, a determinação da corretude de uma célula pode ser suficiente tanto quanto a correção completa de uma célula

Figura 9 – Resultados obtidos para o conjunto de dados Tax_20k. Feramentas (de cima para baixo): Raha, ED2, Trifacta Data Wrangler, DBoost, HoloClean. (Fonte: Elaboração própria).

em termos de desempenho de detecção.

Quanto ao item b), o motivo está atrelado a maneira que os algoritmos decidem o conjunto de tuplas a serem analisadas para o aprendizado de máquina. Enquanto o Raha escolhe as próximas tuplas baseando-se na saída de um conjunto de detectores e algoritmos próprios, o ED2 calcula as próximas células baseando-se em algoritmos de certeza mínima. Além disso, a correção do ED2 é feita célula a célula, enquanto a correção do Raha é feita tupla a tupla, o que deixa o espaço de pesquisa/correção menor.

O Trifacta é uma das ferramentas testadas no estudo de Abedjan et. al [4]. O estudo em questão analisa o desempenho de várias ferramentas baseando-se puramente na troca de conjuntos de dados e comparações de desempenho entre os conjuntos e ferramentas. O estudo demonstra que a capacidade de detecção da ferramenta Trifacta não

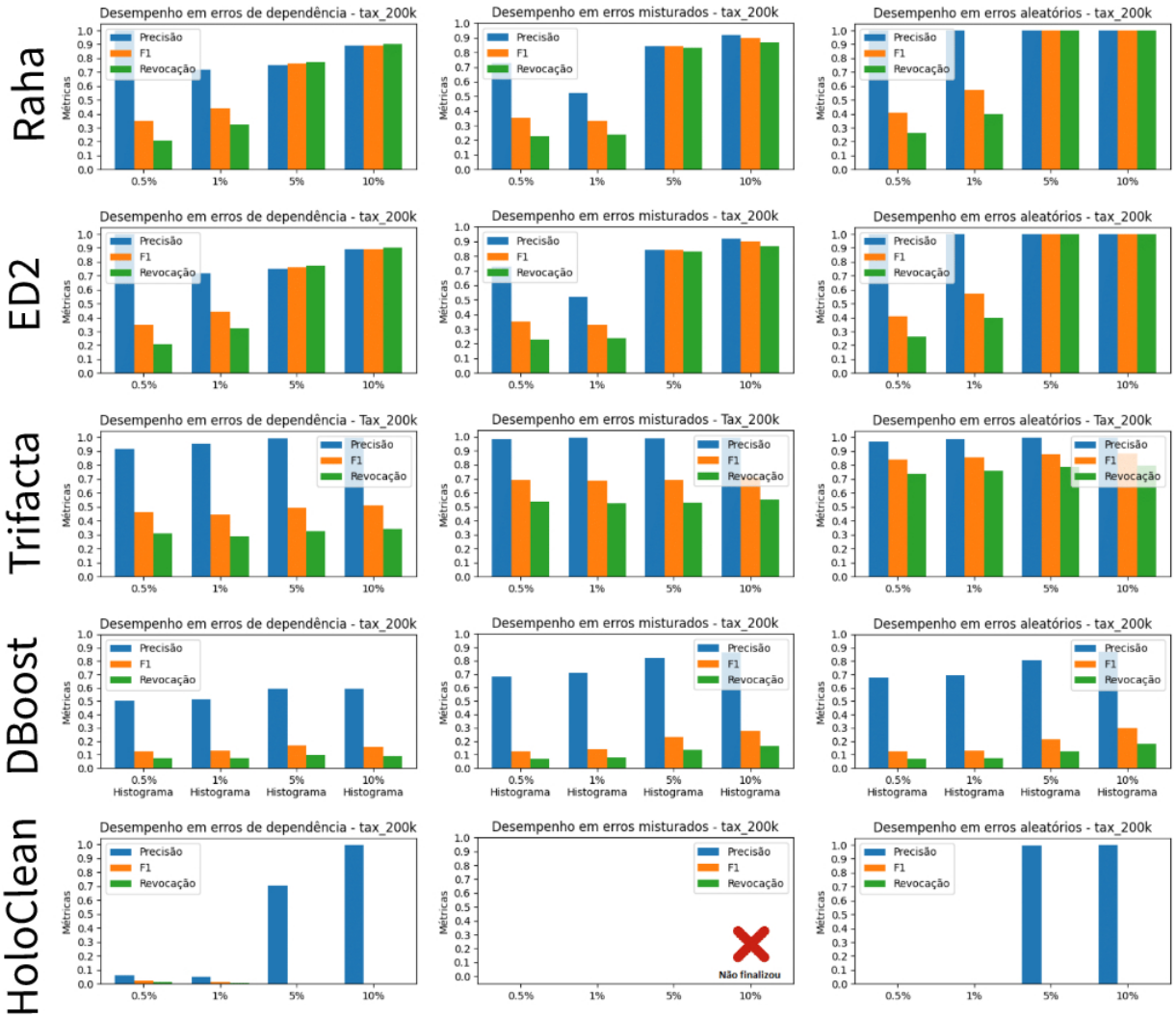

Figura 10 – Resultados obtidos para o conjunto de dados Tax_200k. Feramentas (de cima para baixo): Raha, ED2, Trifacta Data Wrangler, DBoost, HoloClean. (Fonte: Elaboração própria).

foi boa, e não passou de medidas F1 de 50%. O presente estudo demonstrou que nem sempre a ferramenta é uma opção ruim. Particularmente, os testes executados demonstram que taxas de até 90% foram atingidas para erros aleatórios. O alto desempenho dessa ferramenta em testes feitos por este trabalho provavelmente se deve ao fato que a tática de inserções aleatórias feita pelo BART parece consistir unicamente na inserção de erros básicos de digitação. Assim, algoritmos n-grama como o utilizado pelo Trifacta conseguem observar e categorizar com excelência estas pequenas variações. Por se tratar de erros de tipos específicos, a alteração percentual de erros não parece gerar mudanças na capacidade de detecção de erros da ferramenta. Este comportamente é esperado, pois independente da quantidade de erros, um algoritmo n-grama sempre detectará o mesmo tipo de erro.

O DBoost não obteve bom desempenho de detecção. Mesmo com os parâmetros

recomendados, apenas a técnica de histograma conseguiu resultados melhores, e ainda assim longe de satisfatórios. Parte disso se deve ao fato que é difícil estabelecer parâmetros ideais de distribuição gaussiana ou de histogramas a serem considerados sem fazer suposições prévias sobre o conjunto de dados analisado. Por exemplo, considerando que a tática utilizada seja a de histogramas, a ferramenta DBoost pede ao usuário uma taxa de proporção s para consideração. Valores abaixo dessa proporção no histograma são considerados como erros pela ferramenta. Dependendo do conjunto de dados, as taxas ideais podem variar significativamente.

O HoloClean, no estudo original da ferramenta, apresenta medidas F1 excelentes para erros variados. Esta observação não parece ter se sustentado em testes no presente trabalho, em especial para erros de dependência. Mesmo com altas porcentagens de erros, a medida F1 não chegou a 60%. Provavelmente, isso se deve ao fato que a ferramenta utiliza-se do valor de outras células vizinhas às células consideradas como erro. Devido à distribuição de erros utilizada para a inserção, é provável que uma mesma tupla possua vários erros, dificultando o processo de análise de possíveis correções.

O fluxograma, abaixo, apresenta uma ideia geral de qual ferramenta deve ser utilizada para determinada distribuição de erros. O diagrama não deve ser levado como verdade absoluta, mas é um bom indicativo de qual ferramenta funcionará melhor para determinado conjunto a partir dos resultados obtidos pelo presente estudo.

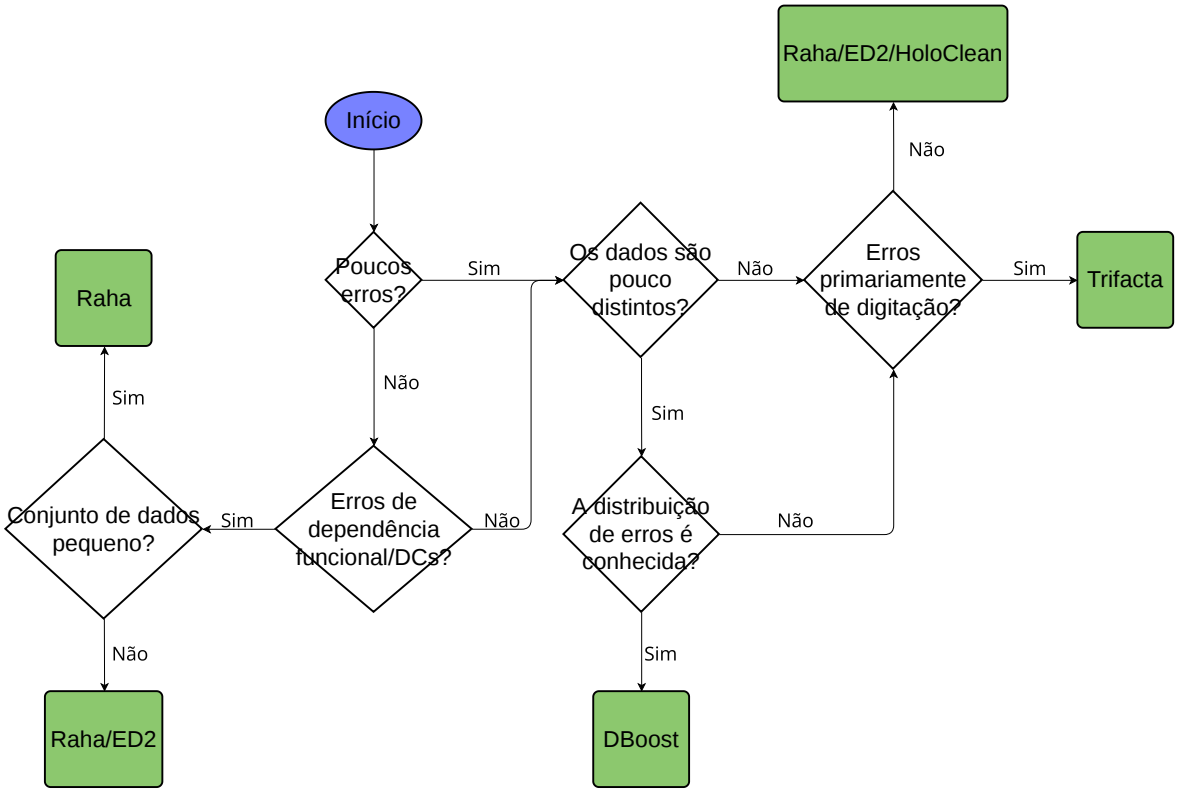

Figura 11 – Fluxograma de escolha de ferramentas. (Fonte: Elaboração própria).

5 CONCLUSÃO

Este trabalho apresentou uma análise dos efeitos que variações específicas e controladas em conjuntos de dados podem causar ao desempenho de detecção de cada ferramenta de limpeza de dados. Com a introdução de diferentes tipos de erros e em diferentes porcentagens, foi possível identificar que soluções novas baseadas em aprendizado de máquina nem sempre são a melhor alternativa para um conjunto de dados. Particularmente, quanto menor a porcentagem de erros ou maior a divergência entre os dados de um conjunto, pior tende a ser o desempenho dessas soluções.

Apesar disso, alternativas recentes apresentam excelente desempenho para conjuntos de dados que possuem erros estruturais (dependências funcionais, dependências funcionais condicionais ou DCs). O grande desafio é a limitação de memória computacional para a utilização dessas soluções.

Para pesquisas futuras, pretende-se inserir erros de forma desequilibrada, onde a distribuição de falhas para determinada coluna será maior ou menor do que outras. Além disso, novas alternativas de corretores de erros (como o Baran [26]) deverão ser testados.

REFERÊNCIAS

[1] ILYAS, I. F.; CHU, X. *Data Cleaning*. New York, NY, USA: Association for Computing Machinery, 2019. ISBN 9781450371520.

[2] KRISHNAN, S. et al. Activeclean: Interactive data cleaning for statistical modeling. *Proc. VLDB Endow.*, VLDB Endowment, v. 9, n. 12, p. 948–959, aug 2016. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/2994509.2994514>.

[3] MARIET, Z. et al. Outlier detection in heterogeneous datasets using automatic tuple expansion. 2016.

[4] ABEDJAN, Z. et al. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.*, VLDB Endowment, v. 9, n. 12, p. 993–1004, aug 2016. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/2994509.2994518>.

[5] HEIDARI, A. et al. Holodetect: Few-shot learning for error detection. In: *Proceedings of the 2019 International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2019. (SIGMOD '19), p. 829–846. ISBN 9781450356435. Disponível em: <https://doi.org/10.1145/3299869.3319888>.

[6] RATNER, A. et al. Snorkel: Rapid training data creation with weak supervision. In: NIH PUBLIC ACCESS. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. [S.l.], 2017. v. 11, n. 3, p. 269.

[7] RÉ, C. Software 2.0 and snorkel: beyond hand-labeled data. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2018. p. 2876–2876.

[8] MAHDAVI, M. et al. Raha: A configuration-free error detection system. In: *Proceedings of the 2019 International Conference on Management of Data*. [S.l.: s.n.], 2019. p. 865–882.

[9] NEUTATZ, F.; MAHDAVI, M.; ABEDJAN, Z. Ed2: A case for active learning in error detection. In: *Proceedings of the 28th ACM international conference on information and knowledge management*. [S.l.: s.n.], 2019. p. 2249–2252.

[10] BEN-GAL, I. Outlier detection. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2005. p. 131–146.

[11] REKATSINAS, T. et al. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, VLDB Endowment, v. 10, n. 11, p. 1190–1201, aug 2017. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3137628.3137631>.

[12] HAM, K. Openrefine (version 2.5). http://openrefine. org. free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA*, Medical Library Association, v. 101, n. 3, p. 233, 2013.

[13] ABEDJAN, Z. et al. Data profiling. *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers, v. 10, n. 4, p. 1–154, 2018.

[14] CHU, X.; ILYAS, I. F.; PAPOTTI, P. Holistic data cleaning: Putting violations into context. In: IEEE. *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. [S.l.], 2013. p. 458–469.

[15] BERTI-EQUILLE, L. et al. Discovery of genuine functional dependencies from relational data with missing values. In: *The 44th International Conference on Very Large Data Bases (VLDB)*. [S.l.: s.n.], 2018. v. 11, n. 8.

[16] MANNILA, H.; RÄIHÄ, K.-J. Algorithms for inferring functional dependencies from relations. *Data & Knowledge Engineering*, Elsevier, v. 12, n. 1, p. 83–99, 1994.

[17] SAXENA, H.; GOLAB, L.; ILYAS, I. F. Distributed discovery of functional dependencies. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. [S.l.: s.n.], 2019. p. 1590–1593.

[18] PAPENBROCK, T.; NAUMANN, F. A hybrid approach to functional dependency discovery. In: *Proceedings of the 2016 International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2016. (SIGMOD '16), p. 821–833. ISBN 9781450335317. Disponível em: <https://doi.org/10.1145/2882903.2915203>.

[19] CHU, X. et al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. [S.l.: s.n.], 2015. p. 1247–1261.

[20] BOHANNON, P. et al. Conditional functional dependencies for data cleaning. In: IEEE. *2007 IEEE 23rd international conference on data engineering*. [S.l.], 2007. p. 746–755.

[21] FAN, W. et al. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering*, v. 23, n. 5, p. 683–698, 2011.

[22] CHU, X.; ILYAS, I. F.; PAPOTTI, P. Discovering denial constraints. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 6, n. 13, p. 1498–1509, 2013.

[23] PENA, E. H.; ALMEIDA, E. C. de; NAUMANN, F. Discovery of approximate (and exact) denial constraints. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 13, n. 3, p. 266–278, 2019.

[24] BLEIFUSS, T.; KRUSE, S.; NAUMANN, F. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 11, n. 3, p. 311–323, 2017.

[25] CARLEO, G. et al. Machine learning and the physical sciences. *Reviews of Modern Physics*, APS, v. 91, n. 4, p. 045002, 2019.

[26] MAHDAVI, M.; ABEDJAN, Z. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 13, n. 12, p. 1948–1961, 2020.

[27] HUANG, Z.; HE, Y. Auto-detect: Data-driven error detection in tables. In: *Proceedings of the 2018 International Conference on Management of Data*. [S.l.: s.n.], 2018. p. 1377–1392.

[28] Trifacta Trifacta Data Wrangler. <https://trifacta.com/>. Accessed: 2022-04-27.

[29] CAVNAR, W. B.; TRENKLE, J. M. et al. N-gram-based text categorization. In: CITESEER. *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*. [S.l.], 1994. v. 161175.

[30] SEUNG, H. S.; OPPER, M.; SOMPOLINSKY, H. Query by committee. In: *Proceedings of the fifth annual workshop on Computational learning theory*. [S.l.: s.n.], 1992. p. 287–294.

[31] KOLLER, D.; FRIEDMAN, N. *Probabilistic graphical models: principles and techniques*. [S.l.]: MIT press, 2009.

[32] AROCENA, P. C. et al. Messing up with bart: error generation for evaluating data-cleaning algorithms. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 9, n. 2, p. 36–47, 2015.

Apêndices

APÊNDICE A – DECISÃO DE PARÂMETROS

Sistemas de detecção de erros que se utilizam de técnicas de aprendizado de máquina exigem parâmetros adicionais. Foi utilizada a ferramenta Raha para testar parâmetros possíveis para os experimentos executados. O gráfico a seguir demonstra o resultado de testes de variação executados na ferramenta.

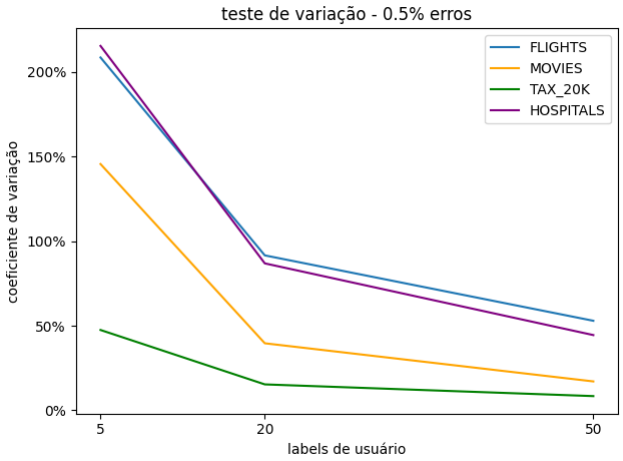

Figura 12 – Testes de variação na ferramenta Raha. (Fonte: Elaboração própria).

A medida que se aumenta a quantidade de rótulos de usuário, o benefício do aumento de rótulos de usuário decresce em ritmo acelerado, sugerindo um gráfico em formato logarítmico. O coeficiente de variação também atinge valores próximos de baixos quando a quantidade de rótulos de usuário atinge 50. Por conseguinte, foi decidido pelo orçamento de rotulagem de 50 rótulos. Este valor também serviu como base para outras ferramentas.