
UNIVERSIDADE FEDERAL DO AMAZONAS

INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

# UMA ESTRATÉGIA EFICIENTE DE TREINAMENTO PARA PROGRAMAÇÃO GENÉTICA APLICADA A DEDUPLICAÇÃO DE REGISTROS.

Manaus

Maio de 2016


UNIVERSIDADE FEDERAL DO AMAZONAS

INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DAVI GUIMARÃES DA SILVA

# UMA ESTRATÉGIA EFICIENTE DE TREINAMENTO PARA PROGRAMAÇÃO GENÉTICA APLICADA A DEDUPLICAÇÃO DE REGISTROS.

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Ciências Exatas da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Prof.Altigran Soares da Silva D.Sc.

Manaus

Maio de 2016

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Silva, Davi Guimarães da
S586u Uma Estratégia Eficiente de Treinamento para Programação Genética Aplicada a Deduplicação de Registros. / Davi Guimarães da Silva. 2016
79 f.: il. color; 31 cm.

Orientador: Altigran Soares da Silva
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Deduplicação. 2. Agrupamento. 3. Programação Genética. 4. Aprendizagem de Máquina. I. Silva, Altigran Soares da II. Universidade Federal do Amazonas III. Título

# FOLHA DE APROVAÇÃO

## "Uma Estratégia de Treinamento para a Programação Genética Aplicada a Deduplicação de Registros"

## DAVI GUIMARÃES DA SILVA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Altigran Soares da Silva - PRESIDENTE

Prof. David Braga Fernandes de Oliveira - MEMBRO EXTERNO

Prof. Moisés Gomes de Carvalho - MEMBRO EXTERNO

Profa. Renata de Matos Galante - MEMBRO EXTERNO

Manaus, 03 de Agosto de 2016

*Aos meus pais, esposa e irmãs.*

# Agradecimentos

Em primeiro lugar quero agradecer a Deus, meu Grande Mestre, que sempre guiou meus passos, me dando sabedoria e entendimento para superar mais este desafio.
Em especial, gostaria de agradecer as seguintes pessoas:
Aos meus pais, Abiezer Pereira e Helena Guimarães, pelo esforço incondicional durante toda a minha caminhada. Por suas orações, e pelas palavras de incentivo nos momentos mais difíceis. Amo vocês!
À minha esposa Nathália Araújo por dividir comigo momentos de alegria e alguns também difíceis. Te amo!
À minhas irmãs, Ábida, Elane e Laiz, pelo apoio e todos os momentos que precisei e por serem parte da minha vida. Meus cunhados, Ronildo, Marques Jr., Jaime Júnior, Edilson, Natanael. Meus sobrinhos Lucas, Emanuelly e Esther, que tanto amo. Ao meu sogro Amarilson e minha sogra Antonia.
Ao meu orientador Prof. Dr. Altigran Soares da Silva e o co-orientador Prof. Dr. Moisés Gomes de Carvalho, por acreditarem, apoiarem, incentivarem, e pelas contribuições fundamentais no desenvolvimento desta dissertação.
Aos professores do ICOMP-UFAM: David Braga Fernandes, pela oportunidade. André Carvalho e Moisés Carvalho, por me possibilitarem a experiência de participar de um projeto inovador. Aos excelentes professores com os quais estudei: Altigran Soares, David Braga, Eulanda Miranda, Marco Cristo, João Cavalcanti, Eduardo Souto.
Ao Instituto Federal do Pará (IFPA), por possibilitar aos seus docentes uma melhor qualificação. Em especial à prof. Dra. Liz Carmem Silva Pereira, Coordenadora de Pesquisa, Pós-Graduação e Inovação do Campus Itaituba.
Ao meu colega de trabalho e de mestrado, Michel Yvano pelo apoio sempre.
Aos meus colegas de projeto, em especial ao Davi Kallebe pelas ideias e ajuda em vários momentos. Ao Duivilly Brito e à Luisa Reis pelas dicas. Vocês serão certamente carregadas para toda a vida.

Emfim, meus mais sinceros agradecimentos a todos que me apoiaram durante esta longa caminhada. Deus os abençoe!

*"Imagino Deus como um programador que conseguiu desenvolver um programa muito interessante chamado HUMANO. O que diferencia este programa de qualquer outro é a quantidade incalculável de variáveis possíveis."*

(Sidney Saymon)

# Resumo

O volume de informação em formato digital tem aumentado consideravelmente nas últimas décadas, e isso tem causado preocupação entre os administradores de grandes repositórios de dados. Trabalhar com esse crescimento e proteger os dados de forma eficaz é um desafio ainda maior. Em muitos repositórios, o principal problema é a existência de dados replicados. Isso pode afetar a qualidade dos dados e a capacidade de fornecer serviços que atendam as demandas dos seus clientes. Porém, a remoção de registros replicados é uma tarefa que exige muito tempo e poder de processamento computacional.

Atualmente, uma das técnicas que vem sendo utilizada de forma eficaz no processo de remoção de registros replicados é a Programação Genética (PG). Uma das principais características dessa técnica é que ela exige exemplos para a realização da etapa de treinamento. Outra característica importante é que a PG exige um alto custo computacional para ser aplicada, além do esforço para gerar os exemplos do treino. No problema de deduplicação um dos maiores custos durante a etapa de treino é causado pela necessidade de comparar cada um dos registros com todos os outros registros existentes no banco de dados. Assim, o tempo gasto para realizar essas comparações durante o treino é muito grande.

A partir desse problema, esta dissertação propõe uma abordagem baseada na combinação de uma técnica de agrupamento e janela deslizante, visando minimizar a quantidade de comparações exigidas na etapa de treinamento da PG. Experimentos utilizando dados reais e sintéticos, mostram que é possível reduzir o custo de treinamento em até 70%, sem uma redução significativa na qualidade das soluções geradas.

**Palavras-chave:** Deduplicação, Agrupamento, Programação Genética, Aprendizagem de Máquina.

# Abstract

The amount of information available through digital media has increased considerably in recent decades. This fact causes concern among managers of large data repositories. Dealing with this growth and protect the data effectively is an even greater challenge. In many repositories, one of the main problems is the existence of replicated data. This can impact the quality of data and the ability to provide services able to meet the demands of its customers. However, the removal of replicated records is a task that requires a lot of time and processing effort.

Nowadays, one of the techniques that has been effectively applied in the task of identify records that are replicated is the Genetic Programming (GP). One of the main requirements of this technique is the use examples (usually created manually) in its training step. Another GP major requirement is its processing time. This happens because during the training step each record is compared to all other existing ones in the data repository. Thus, the time required to perform all these comparisons during the GP training step can be very costly, even for small repositories.

For those reasons, this dissertation proposes a novel approach based in a strategy the combines a clustering technique with a sliding window, aiming at minimize the number of comparisons required in the PG training stage. Experiments using synthetic and real datasets show that it is possible to reduce the time cost of GP training step up to 70%, without a significant reduction in the quality of generated solutions.

**Keywords:** Deduplication, Clustering, Genetic Programming, Machine Learning, blocking.

# Lista de Figuras


3.1 Exemplo de indivíduo em PG [Carvalho et al., 2008a]. . . . . 18
3.2 Cruzamento em PG [Carvalho et al., 2008a]. . . . . 20
3.3 Mutação em PG [Carvalho et al., 2008a]. . . . . 21
3.4 Fluxograma do funcionamento básico de PG [Carvalho et al., 2008a]. . . . 23

4.1 Esquema de Deduplicação utilizando PG [Gonçalves, 2009]. . . . . 32
4.2 Exemplo de registros e seus atributos. . . . . 33
4.3 Exemplo de pares de registros em um repositório. . . . . 36

5.1 Exemplo de *String ruído*. . . . . 43
5.2 Divisão de um arquivo em *clusters*. . . . . 45

6.1 Exemplo da estratégia de janela deslizante. . . . . 51

# Lista de Tabelas


5.1 Características dos *datasets* utilizados em nossos experimentos. . . . . 40
5.2 Aplicação das funções de similaridade no *dataset Restaurants*. . . . . 46
5.3 Aplicação das Funções de similaridade no *dataset Cora*. . . . . 46
5.4 Aplicação das Funções de similaridade no *dataset Synthetic*. . . . . 47

6.1 Principais parâmetros de PG utilizados. . . . . 51
6.2 Configurações da Janela e Deslocamento . . . . . 52
6.3 Configurações do computador. . . . . 53
6.4 Resultados dos experimentos no *dataset Restaurants* . . . . . 54
6.5 Resultados dos experimentos no *dataset Restaurants* . . . . . 55
6.6 Resultados dos experimentos no *dataset Synthetic* . . . . . 56
6.7 Resultados dos experimentos no *dataset Synthetic* . . . . . 57
6.8 Resultados dos experimentos no *dataset Cora* . . . . . 58
6.9 Resultados dos experimentos no *dataset Cora* . . . . . 59

# Sumário

**Agradecimentos** **vi**

**Resumo** **viii**

**Abstract** **ix**

**Lista de Figuras** **x**

**Lista de Tabelas** **xi**

**1 Introdução** **1**
1.1 Motivação . . . . . 2
1.2 Objetivos . . . . . 3
1.3 Contribuições . . . . . 4
1.4 Organização da Dissertação . . . . . 4

**2 Trabalhos Relacionados** **6**
2.1 Abordagens baseadas em Aprendizado de Máquina para Deduplicação . 6
2.2 Técnicas de Agrupamento para Deduplicação . . . . . 9
2.3 Técnicas de Blocagem para Deduplicação . . . . . 10
2.4 Técnicas para Seleção de Exemplos de Treino . . . . . 13

**3 Conceitos Básicos** **15**
3.1 Aprendizagem de Máquina . . . . . 15
3.1.1 Treinamento no Aprendizado Supervisionado . . . . . 16
3.2 Programação Genética . . . . . 16
3.2.1 Indivíduo em PG . . . . . 17
3.2.2 Operadores Genéticos . . . . . 19
3.2.3 Processo Evolucionário . . . . . 22
3.2.4 Etapa de Treinamento em PG . . . . . 24

3.3 Agrupamento de Dados . . . . . 24
3.3.1 Métodos de Agrupamento . . . . . 25

**4 Deduplicação de Registros com PG** **29**
4.1 Deduplicação de Registros . . . . . 29
4.2 Modelagem para Problema de Deduplicação de Registros com PG . . . 30
4.2.1 Definição sobre Pares de Registros . . . . . 33
4.2.2 Função de *Fitness* na Deduplicação . . . . . 34
4.2.3 Complexidade da Etapa de Treinamento . . . . . 35
4.2.4 Considerações sobre a Etapa de Treinamento . . . . . 36

**5 Técnica de Agrupamento para a Etapa de Treino em PG** **38**
5.1 Fundamentação para a Abordagem Proposta . . . . . 38
5.2 Abordagem para Técnica de Agrupamento . . . . . 39
5.2.1 Abordagem Experimental . . . . . 39

**6 Estratégia para Comparação de Registros** **49**
6.1 Estratégia de Janela Deslizante . . . . . 49
6.2 Configuração Experimental . . . . . 51
6.3 Métricas de Avaliação . . . . . 53
6.4 Avaliação Experimental da Técnica Proposta . . . . . 53
6.4.1 Experimentos com o *dataset Restaurants* . . . . . 54
6.4.2 Experimentos com o *dataset Synthetic* . . . . . 55
6.4.3 Experimentos com o *dataset Cora* . . . . . 57
6.5 Considerações Finais do Capítulo . . . . . 59

**7 Conclusões e Trabalhos Futuros** **61**
7.1 Conclusões . . . . . 61
7.2 Trabalhos Futuros . . . . . 62

**Referências Bibliográficas** **63**

# Capítulo 1

# Introdução

O volume de dados gerados por usuários dos mais variados tipos de sistemas tem aumentado de forma significativa a cada ano. Uma pesquisa elaborada pela IDC em 2011, [Gantz, 2012], aponta que o volume de dados está mais do que dobrando a cada dois anos e deve atingir 11,8 zettabytes (1,8 trilhões de gigabytes) em pouco tempo. De acordo com uma pesquisa realizada pela IBM[1] aproximadamente 90% dos dados armazenados no mundo foram criados nos últimos dois anos. Outras pesquisas, [Woolf, 2010] e *The Economist Newspaper Limited*[2], revelam que 30 bilhões de publicações são compartilhadas no Facebook[3] por mês. E em 2020, as empresas terão que administrar 10 vezes mais servidores, 50 vezes mais dados, 75 vezes mais arquivos, com apenas 1,5 vezes mais pessoas.

Esse crescimento se dá pelas possibilidades de compartilhamento de informações por meios digitais independente da localização geográfica do usuário. Devido ao aumento da quantidade desses dados, administradores de grandes repositórios de dados, como bibliotecas digitais e bancos de dados de grandes empresas, vêm encontrando problemas no que se refere à qualidade dos dados.

Geralmente, os repositórios de dados são construídos a partir da junção entre diferentes fontes de dados. Assim, é possível que ocorram diversas inconsistências, permitindo a geração indesejada de dados "sujos". Esses dados "sujos" se referem a algum registro[4] que possui descrição imprecisa, incompleta ou errônea, e que pode ser encontrado mais de uma vez no repositório de dados.

De acordo com [Gonçalves, 2009], é possível estabelecer uma relação entre a qualidade dos dados presentes nos sistemas de uma organização e sua capacidade de prover

[1] http://www-01.ibm.com/software/data/bigdata
[2] http://www.economist.com/node/15557443
[3] https://www.facebook.com
[4] é uma instância de uma tabela, ou entidade. É constituído por conjunto de campos valorados.

serviços de qualidade a seus clientes. A decisão de manter esses dados "sujos", afeta a performance e desempenho dos sistemas que os utilizam, demandam mais tempo para responder consultas simples feitas pelos usuários, além de gerar distorções em relatórios, levando à tomada de decisões incorretas.

O problema de detecção e remoção de registros repetidos em um repositório é geralmente conhecido como deduplicação de registros [Koudas et al., 2006]. Na literatura também pode ser descrito como limpeza de dados [Chaudhuri et al., 2003], pareamento de registros ([Bhattacharya, 2004], [Fellegi, 1969], [Koudas et al., 2006]) e casamento de registros [Verykios et al., 2003].

A deduplicação de registros em bancos de dados consiste em identificar e remover registros que são potencialmente os mesmos em uma base de dados (ou em um conjunto de bases de dados), ainda que tenham estilos de escrita, grafias, tipos de dados, e até esquemas diferentes. É uma tarefa complexa, cujo tratamento requer muito tempo e poder de processamento devido à grande quantidade de comparações necessárias para definir se um registro possui uma ou mais réplicas no banco de dados.

## 1.1 Motivação

Nos últimos anos, diversos métodos foram propostos para remoção de réplicas em grandes repositórios de dados. Recentemente, [Carvalho et al., 2008a] apresentaram uma abordagem inovadora para a identificação de registros duplicados em repositórios de dados, recorrendo a uma técnica de aprendizado de máquina conhecida como Programação Genética (PG) [Koza, 1992] [Banzhaf et al., 1998]. Essa abordagem é atualmente o estado da arte em deduplicação de registros por apresentar resultados superiores a outras abordagens encontradas na literatura.

A utilização dessa técnica tem como ponto positivo a alta precisão no processo de deduplicação de bases de dados que possuem diferentes características. Como ponto negativo pode se destacar o alto custo computacional da PG. Isso ocorre porque cada registro é comparado com todos os outros registros existentes no banco de dados. Assim, o tempo gasto para realizar essas comparações é muito grande, o que torna pouco viável a adoção desta técnica.

Existem muitas abordagens que visam diminuir o custo do processamento de PG propondo a redução do tempo exigido na etapa de treinamento. Essa etapa visa "ensinar" a técnica de aprendizagem a identificar as características relevantes de boas soluções (que são dadas como exemplos). Se o tempo gasto na etapa de treinamento fosse reduzido, essa técnica se tornaria mais facilmente adotada.

Com base nesses dados coletados, este trabalho propõe um método baseado na combinação de uma técnica de agrupamento e janela deslizante [Ziv et al., 1977], para minimizar a quantidade de comparações que são exigidas na etapa de treinamento de PG. Com esta abordagem, pode se reduzir o custo de treinamento em até 70%, mantendo o mesmo nível de qualidade das soluções obtidas pela abordagem de [Carvalho et al., 2008a].

A técnica proposta busca minimizar a quantidade de comparações feitas em exemplos de treino com pares de registros negativos, isto é, registros que não fazem referência ao mesmo objeto do mundo real.

A principal motivação para este trabalho surgiu a partir de um projeto de pesquisa realizado por dois professores da Universidade Federal do Amazonas (UFAM) em parceria com a Samsung Eletrônica da Amazônia Ltda. Como parte do projeto, foi realizada a coleta de aproximadamente 850.000 (oitocentos e cinquenta mil) dados de Objetos de Aprendizagem[5] (OAs) existentes na web provenientes de diferentes repositórios.

Por se tratar de dados coletados em diferentes repositórios, observou-se a existência de OAs replicados. Assim, visando minimizar esse problema, tornou-se necessário utilizar uma abordagem para encontrar os OAs replicados. Tendo em vista que a abordagem de [Carvalho et al., 2008a] obteve melhores resultados na identificação de registros replicados em comparação a outras abordagens existentes na literatura, decidimos utilizá-la.

Porém, conforme já mencionado, o método de [Carvalho et al., 2008a] demanda muito tempo para realizar a etapa de treinamento o que tornaria esse processo bastante custoso. Com base isso, utilizando como arcabouço o método de [Carvalho et al., 2008a], propomos a abordagem citada acima para minimizar o tempo gasto na etapa de treinamento, sem comprometer significativamente as soluções obtidas por eles.

## 1.2 Objetivos

*Geral:*
Tornar a etapa de treino da técnica de PG para uma abordagem de deduplicação de registros mais rápida sem comprometer sua eficácia.

[5]De acordo com o *Learning Objects Metadata Workgroup* são "qualquer entidade, digital ou não digital, que possa ser utilizada, reutilizada ou referenciada durante o aprendizado suportado por tecnologias".

*Específicos:*

- Elaborar uma estratégia para o uso de técnica de agrupamento com o objetivo de agrupar registros por similaridade;
- Elaborar uma técnica de janela deslizante aplicada em conjunto com a técnica de agrupamento para a etapa de treinamento de PG.

## 1.3 Contribuições

As principais contribuições desta dissertação são:

1) Uma técnica para redução do tempo de treino aplicado a uma abordagem baseada em PG, que:
   - É efetiva quanto a diminuição do tempo utilizado na etapa de treino quando comparado ao método estado da arte [Carvalho et al., 2008a] (utilizado como arcabouço nesta proposta), mantendo um nível de qualidade similar as soluções originais.
   - Fornece solução com menor custo computacional, uma vez que a quantidade de comparações entre registros na etapa de treino será reduzida consideravelmente por meio do uso da técnica de agrupamento.
2) Contribuições gerais:
   - Propor uma nova forma de utilizar a técnica de janela deslizante, aplicada na etapa de treinamento da PG.
   - Apresentar os impactos no desempenho do treino da PG.
   - Provê diretrizes para definição dos parâmetros utilizados para os testes.

## 1.4 Organização da Dissertação

Esta dissertação está estruturada da seguinte forma:

- No Capítulo 2, é apresentado um estudo bibliográfico relativo aos trabalhos relacionados a utilização das técnicas de agrupamento para deduplicação de registros, e abordagens relacionadas a etapa de treino em PG, além de uma análise dos trabalhos relacionados.

- No Capítulo 3, são tratados os conceitos básicos sobre aprendizagem de máquina. É descrito também pontos importantes sobre PG e sua aplicação no contexto do problema de deduplicação de registros; além dos principais fundamentos das técnicas de agrupamento.

- No Capítulo 4, é abordada a utilização da técnica de PG aplicada ao contexto do problema de deduplicação de registros.

- No Capítulo 5 é descrita a abordagem proposta que utiliza a técnica de agrupamento melhorar a performance da estratégia de janela deslizante descrita no capítulo seguinte. São apresentados também experimentos para escolha da função mais efetiva para realizar o agrupamento de registros originais próximos de suas réplicas.

- No Capítulo 6, é descrita a adoção de uma estratégia de janela desliante. Além disso, é promovida uma extensiva avaliação experimental a fim de validar as propostas apresentadas por esta dissertação. Esse capítulo tem como objetivo avaliar e validar a proposta apresentada no Capítulo 5.

- No Capítulo 7 são apresentadas as conclusões com base nos resultados obtidos e as possíveis direções de pesquisa a serem desenvolvidas no futuro.

# Capítulo 2

# Trabalhos Relacionados

Este capítulo apresenta um estudo bibliográfico relativo aos trabalhos relacionados ao escopo desta dissertação. Na seção 2.1, são apresentadas as principais abordagens de aprendizagem de máquina para deduplicação. Na seção 2.2 são tratados os trabalhos que utilizam técnicas de agrupamento para deduplicação. Já na seção 2.4 é descrita uma técnica que utiliza seleção de exemplos para o treino da PG.

## 2.1 Abordagens baseadas em Aprendizado de Máquina para Deduplicação

Não é de hoje que trabalhos têm proposto métodos baseados em técnicas de aprendizado de máquina para solucionar o problema da deduplicação. A primeira proposta para esse problema foi apresentada por [Newcombe et al., 1959]. No entanto, o primeiro modelo formal consolidado na bibliografia para deduplicação de registros foi de [Fellegi, 1969], tendo como base a proposta esboçada por [Newcombe et al., 1959]. O modelo de [Fellegi, 1969] é baseado em probabilidades, que visa otimizar a classificação dos pares de duas bases. Nela os autores formularam uma série de teorias para a formalização do problema da deduplicação. Tal embasamento teórico é utilizado até os dias atuais em inúmeros trabalhos científicos, como os que serão abordados a seguir.

[Tejada et al., 2001] apresentam um sistema chamado *Active Atlas*, cujo objetivo principal é realizar o mapeamento entre registros oriundos de diferentes fontes de dados. As regras de mapeamentos de atributos são especificadas com base em um processo de treinamento que utiliza árvores de decisão e aprendizado ativo. Os atributos são comparados utilizando funções que identificam substrings, acrônimos e abreviações. O resultado das funções é combinado por uma modificação da métrica TF x IDF que

pondera as dimensões vetoriais. O processo de combinação dos pesos é executado utilizando árvores de decisão.

[Tejada et al., 2002] apresentam uma estratégia baseada em aprendizado ativo em que, novamente, árvores de decisão são utilizadas no ensino de regras para a deduplicação de registros com múltiplos atributos. O método sugere que, com a criação de múltiplos classificadores, treinados com dados ou parâmetros ligeiramente diferentes, é possível detectar casos ambíguos e então pedir uma resposta por parte do usuário. Segundo [Elmagarmid et al., 2007], a principal inovação desse trabalho está na criação de diversas funções redundantes e na exploração concorrente de suas ações conflitantes, visando a descoberta de novos tipos de inconsistência entre réplicas no conjunto de dados.

[Bilenko, 2003] projetaram um *framework* com intuito de maximizar a eficácia da deduplicação utilizando funções de similaridade adaptáveis ao domínio. A abordagem, chamada MARLIN (*Multiply Adaptive Record Linkage using INduction*) se utiliza de uma amostra, previamente rotulada, para o treinamento em duas camadas: *(i) Nível de atributo*: é proposta a utilização de uma variante da função de distância de edição, na qual são adicionados pesos a substituições, inserções e remoções. O intuito é adaptar o cálculo de similaridade de acordo com o ruído presente na base de dados. *(ii) Nível de registro*: o algoritmo *SVM* é treinado com os vetores de características. Como resultado, é produzido um modelo de treinamento que é responsável pela quantização do grau de confiança do classificador, ou seja, a distância do par à fronteira do hiperplano. Por fim, o grau de confiança é avaliado por um classificador binário.

Nessa perspectiva os experimentos propostos tiveram como objetivo avaliar a eficácia da abordagem, assim como, a adaptação alcançada pelo processo de aprendizagem. MARLIN apresentou uma melhora na eficácia comparada aos métodos tradicionais (puros) de treinamento (por exemplo, SVM). A avaliação foi feita em base de dados reais com cerca de 800 a 1.200 registros. Apesar de sugerir a possibilidade de adoção de algoritmos de aprendizagem ativa para a criação da base de treinamento, nenhum experimento foi conduzido com intuito de avaliar o número de pares necessários para a convergência do processo de treinamento.

Em [Carvalho et al., 2006] apresentam uma abordagem baseada em programação genética (PG) para gerar automaticamente funções de similaridade que identifiquem registros duplicados em um determinado repositório. As funções são representadas por uma expressão matemática que combina e pondera os atributos que formam um registro. Uma expressão é implementada como uma árvore binária cujos nós folha são atributos e os nós intermediários são operações matemáticas. A cada geração de indivíduos, um novo conjunto de árvores é gerado a partir do cruzamento das árvores

da geração anterior. Além de herdar características, um fator de mutação garante um determinado nível de diversidade na população. Os experimentos realizados pelos autores mostram que as funções geradas são mais efetivas para identificar duplicatas que o método estatístico proposto por [Fellegi, 1969].

Para [Carvalho et al., 2008b], a abordagem baseada em programação genética é aplicada de forma independente do método de [Fellegi, 1969]. Uma vez que a identificação de réplicas é uma tarefa que consome muito tempo, mesmo para repositórios de dados pequenos, o método proposto tenta combinar os melhores fragmentos de evidências para a geração de funções de similaridade que maximizem o desempenho, utilizando para isto uma pequena porção do repositórios de dados para treino.

Em um terceiro artigo [Carvalho et al., 2008a] relata um estudo que descreve as principais propriedades da programação genética e aborda detalhes sobre a parametrização do algoritmo [Carvalho et al., 2008b]. Os experimentos apresentados mostram que a escolha de alguns parâmetros pode melhorar o resultado da deduplicação em até 30% comparado ao resultado da aplicação anterior.

[Freitas et al., 2010] propõem o método *Active GP* para deduplicação baseado em programação genética semisupervisionada, apropriado para os casos em que o esforço humano para rotular um conjunto de treinamento é muito alto. Este método utiliza aprendizagem ativa, na qual um comitê de funções de deduplicação multiatributo vota para classificar pares como réplicas ou registros distintos. Os resultados dos experimentos apresentados mostram que, quando comparado ao método proposto por [Carvalho et al., 2006], a qualidade da deduplicação se mantém, reduzindo consideravelmente o número de instâncias de treinamento rotuladas.

Recentemente, [Carvalho et al., 2012] generalizam os resultados do método proposto mostrando que programação genética também é capaz de encontrar funções de deduplicação efetivas, mesmo quando as funções de similaridade adequadas para cada atributo não são conhecidas de antemão. Esta característica é extremamente útil para o usuário porque ele não precisa se preocupar com a seleção de funções e análise de domínio dos atributos. Além disso, os autores demonstram que o método proposto adapta a função de deduplicação de acordo com os limiares de similaridade escolhidos para classificar um par como réplica ou registros distintos. Essa característica libera o usuário do ônus causado pela sintonia fina desses limiares.

[Dal Bianco et al., 2013] propõem um arcabouço chamado FS-Dedup que auxilia no desempenho e qualidade do processo de deduplicação de grandes volumes de dados que dependem de usuários especialistas para configurar as fases mais importantes no processo: a estratégia de blocagem e o algoritmo de classificação. O FS-Dedup explora algoritmos de deduplicação baseada em assinatura.

Esses algoritmos são caracterizados pela alta eficiência e escalabilidade em grandes conjuntos de dados. A abordagem exige que o usuário rotule apenas um pequeno conjunto de pares de registros para sintonizar os parâmetros automaticamente. A partir dos pares rotulados é identificada a região crítica onde pares duplicados e distintos retornam valores de similaridade muito próximos. Os pares desta região são utilizados para configurar modelos de classificação baseados no algoritmo *SVM* e na função n-gram. Os experimentos realizados sobre conjuntos de dados reais e sintéticos contendo milhões de registros mostram que a proposta atinge a qualidade máxima da técnica de deduplicação baseada em assinatura com um esforço manual do usuário bastante reduzido.

[Dal Bianco et al., 2015] os autores propõe uma estratégia de seleção de amostragem em duas fases (T3S) que seleciona um conjunto reduzido de pares para ajustar o processo de deduplicação em grandes conjuntos de dados. T3S seleciona os pares mais representativos, seguindo duas etapas. Na primeira etapa, foi proposta uma estratégia para a produção de pequenas sub-amostras aleatórias de pares candidatos em diferentes frações de conjuntos de dados. Na segunda etapa, sub-amostras são analisadas de forma incremental para remover redundância de pares criadas no primeiro estágio, a fim de produzir um conjunto de treino, mesmo menor e mais informativa. Em seus experimentos, foram avaliados conjuntos de dados reais e sintéticas e empiricamente mostraram que, em comparação com seus *baselines*, o T3S é capaz de reduzir consideravelmente o esforço do usuário, mantendo a mesma ou uma melhor eficácia.

## 2.2 Técnicas de Agrupamento para Deduplicação

Com o objetivo de diminuir o tempo para deduplicação de registros, algumas técnicas normalmente são adotadas em conjunto com as técnicas de deduplicação. A seguir serão descritos alguns trabalhos que utilizaram a técnica de agrupamento para deduplicação de registros.

[Cohen, 2002] apresenta uma técnica escalável e adaptativa para agrupar registros duplicados. A ideia básica é utilizar uma métrica de comparação com baixo custo computacional para agrupar registros. Assim, os pares candidatos ao casamento são drasticamente reduzidos utilizando uma estratégia de blocagem baseada na métrica TF X IDF para múltiplas funções de similaridade de strings. Os registros candidatos ao casamento são organizados como vértices de um grafo em que as arestas representam o fator de confiança na predição do par de vértices ser duplicado. Os grupos são gerados a partir de um algoritmo guloso que considera inicialmente cada vértice um grupo único e

a cada iteração funde os grupos mais próximos até que restem apenas um determinado número de grupos passado como parâmetro.

A abordagem de [Bilenko, 2003] citado na seção anterior, chamada MARLIN, emprega o método de agrupamento *Canopy* proposto por [Mccallum et al., 2000], para promover um processo de blocagem em duas etapas. Na primeira etapa, é utilizada a bem conhecida função de similaridade Jaccard ([Koudas et al., 2006]), com baixo custo computacional. Na segunda etapa, é utilizada uma estrutura básica de índice invertido ([Manning et al., 2008]) para o agrupamento de registros.

Em [Santos et al., 2008], é apresentada uma abordagem para deduplicação com intuito de remover a intervenção do usuário. A abordagem avalia a qualidade interna dos *clusters* para extrair indícios em busca do limiar ideal. É dividida em duas etapas: Na primeira etapa, são gerados grupos a partir de um algoritmo para agrupamento hierárquico chamado HAC proposto por [Manning et al., 2008]. Na segunda etapa, é avaliada a qualidade dos grupos produzidos por cada limiar a partir das medidas de coesão e separação. Para experimentos, foram utilizados duas bases de dados reais e quatro duas bases de dados sintéticas compostas por cerca de 300 a 2.000 registros. Assim, comprovaram experimentalmente que o coeficiente de silhueta proporciona uma alta correlação com a métrica F1, na maioria das bases testadas.

## 2.3 Técnicas de Blocagem para Deduplicação

Vários estudos sobre métodos de blocagem podem ser encontrados na literatura atual. Os métodos mais recentes são baseados na técnica de Aprendizagem de Máquina *(Machine Learning)* para determinar a melhor função de blocagem para o agrupamento ou comparação de registros. Esses métodos são diferentes dos métodos estáticos, uma vez que os estáticos são baseados em regras de agrupamento predefinidos e não mudam de acordo com as características encontradas nos registros.

[Chaudhuri et al., 2003] propõem um algoritmo probabilístico para recuperar os K registros mais similares a um determinado registro de entrada, de acordo com uma função de casamento aproximado *fuzzy*. Essa função considera o peso das palavras contidas nos registros usando a métrica *Inverse Document Frequency (IDF)* [Baeza-Yates, 2011] do modelo espacial vetorial de recuperação de informações. Esta solução foi proposta para comparar registros em *data warehouses*. O parâmetro K pode ser substituído por um limiar de similaridade definido pelo usuário. Para reduzir o número de pares a ser comparado é utilizada blocagem baseada em *q-grams*. A técnica proposta é extensível, pois possibilita o uso de funções de pesos específicas para certos

domínios ao invés dos pesos gerados pela métrica *IDF*.

[Bilenko et al., 2006] sugeriu o método *DNF Blocking*, onde as instâncias de treinamento para aprendizagem de máquina são pares de registros classificados como: verdadeiros, se encontrados na mesma entidade; ou falsos, se os registros são de entidades diferentes. Essas instancias de treinamento são usadas em análises para a escolha de regras que produzam os melhores agrupamentos de registros na blocagem. Essas regras são selecionadas e combinadas para formar expressões na forma normal disjuntiva (FND), chamadas de esquemas de blocagem. O método *DNF Blocking* pode apresentar resultados melhores nos casos em que os métodos estáticos falham. Porém, se aumentar o número dessas regras pode significar um aumento também do tempo necessário para o processo de aprendizagem de máquina, dificultando o uso desse método.

Um método semelhante é proposto por [Michelson, 2006], trata-se do *BSL Blocking*. No que se refere ao uso de regras de blocagem, esse método é semelhante ao *DNF Blocking*, porém as maiores diferenças estão na forma como as amostras de treino são usadas e no método de combinação dos predicados para produzir os esquemas de blocagem nos processos de aprendizagem de máquina. Na pratica, pode ser observado que o método *BSL Blocking* é normalmente usado com amostras contendo números pequenos de pares de registros de treino, por considerar apenas os pares verdadeiros das amostras de treino. Por esse motivo pode apresentar tempos menores de exceção, comparados aos tempos de execução do método *DNF Blocking*. Essa vantagem pode ser usada nos casos em que o tempo curto de processamento for um requisito critico para o agrupamento dos registros.

Em [Evangelista et al., 2009], os autores apresentam um método de blocagem adaptativa baseado em aprendizagem de máquina denominado BGP (Blocagem baseada em Programação Genética). Essa abordagem permite o uso de regras mais flexíveis e um maior numero de regras para a definição de funções de blocagem, aumentando também a eficacia na identificação de registros duplicados. O método BGP, assim como os métodos DNF Blocking e BSL Blocking, são baseados em esquemas de blocagem formados a partir de predicados booleanos para a identificação dos pares de registros que correspondem a mesma entidade no mundo real. Ou seja, o método BGP também requer amostras de dados para descobrir bons esquemas de blocagem, usando aprendizagem de maquina para atingir esse objetivo. Contudo, o método BGP diferencia-se dos métodos baseados em aprendizagem de maquina, no que diz respeito a forma como os predicados de blocagem são combinados no processo de procura pelo melhor esquema de blocagem. Os Resultados de experimentos com dados reais e sintéticos mostram que percentuais de acertos acima de 95% podem ser conseguidos na detecção de pares duplicados de registros de maneira eficiente.

[Vernica et al., 2010] propõe um algoritmo baseado em assinaturas para o processamento em larga escala. O algoritmo utiliza o modelo de programação distribuído MapReduce para a paralelização de tarefas. No MapReduce, o usuário é responsável pela especificação das funções de mapeamento (Map) e redução (Reduce), enquanto o modelo de programação se encarrega de paralelizar e distribuir as tarefas sem a intervenção do usuário. O método proposto é divido em três principais estágios. No primeiro estágio, é criado um ordenamento global dos tokens para a definição das assinaturas de cada registro. No segundo estágio, são aplicados os filtros (prefixo, tamanho, posição e sufixo) sobre as assinaturas. Dessa forma, são criados conjuntos de pares candidatos. No estágio três, os pares de registros são reconstruídos para o salvamento no arquivo de saída. Os experimentos foram executados na base de dados do DBLP e CITESEERX. A experimentação dos autores demonstrou que o método foi capaz de processar uma base de dados com mais de 30 milhões de registros.

[Dal Bianco et al., 2011] apresentam um deduplicador que combina um eficiente método de blocagem com o modelo de programação MapReduce, com intuito de melhorar o desempenho e a qualidade da deduplicação em arquiteturas multiprocessadas. Na abordagem chamada MD-Approach, é proposto um deduplicador que emprega um método de blocagem em duas etapas para evitar o desbalanceamento de carga, ou seja, blocos substancialmente grandes podem consumir recursos (processadores) por um longo período mesmo que os demais blocos tenham sido processados. No MD-Approach, os pares candidatos são criados através da combinação de atributos (ou partes de atributos) para a construção das chaves de blocagem. A blocagem é realizada em duas etapas: A primeira etapa de blocagem tem como objetivo a criação de blocos genéricos para assegurar que registros duplicados não sejam erroneamente removidos do bloco correspondente. Na segunda etapa, os blocos são fragmentados em sub-blocos através da definição de chaves de blocagem mais especializadas (por exemplo, três primeiras letras dos valores de um atributo). Dessa forma, a blocagem em duas etapas fragmenta a base de dados em pequenos blocos minimizando o custo de processamento. Os experimentos foram executados com dados sintéticos variando de um a quatro milhões de registros. Os experimentos demostraram que o MD-Approach foi até duas vezes mais eficiente que o baseline, com uma melhora de mais de 9% no valor do F1. Entretanto, a eficiência da abordagem MD-Approach depende da definição adequada dos valores de limiares.

[Whang, 2012] propõe um método para solucionar o problema da deduplicação conjunta. Os autores especificam um arcabouço modular e flexível em que múltiplos algoritmos de deduplicação podem ser sintonizados para tipos particulares de registros. São definidos planos de execução lógica destes algoritmos que respeitam determinadas

restrições de recursos como quantidade de memória e número de núcleos de CPU. A abordagem proposta encontra o plano de execução mais eficiente dado uma quantidade limitada de recursos. O comportamento e a escalabilidade da abordagem proposta são avaliados a partir de um conjunto de experimentos realizados sobre bases de dados sintéticas e reais. É utilizada a estratégia de blocagem vizinhança ordenada, a função de similaridade Jaro e o algoritmo de deduplicação.

Em um trabalho posterior [Whang, 2013], os autores propõe um *framework* de resolução flexível, modular, onde algoritmos ER existentes desenvolvidos para um determinado tipo de registro pode ser conectado e usado em conjunto com outros algoritmos de ER. A abordagem também faz com que seja possível executar ER em subconjuntos de registos semelhantes de cada vez, importante quando os dados completos são grandes para resolver juntos. Também introduziram uma técnica de treinamento *state-based*, onde cada algoritmo ER é treinado para o contexto de execução particular (em relação a outros tipos de registros), onde ele será usado.

## 2.4 Técnicas para Seleção de Exemplos de Treino

O trabalho mais relacionado ao nosso encontrado na literatura foi o de [Gonçalves, 2009]. Trata-se de uma abordagem que utiliza uma técnica determinística para sugerir automaticamente exemplos de treino para um método de deduplicação de registros baseado em PG. Foi baseada no método proposto por [Carvalho et al., 2008a].

Os passos envolvidos ao se aplicar essa abordagem são:
1) O repositório a ser deduplicado é dividido igualmente em quatro arquivos, sendo um para treino e os demais para avaliação.
2) Aplica o método determinístico de [Fellegi, 1969] que deduplica o conjunto de treino e gera uma lista com todos os pares de registros positivos e outra com todos os pares de registros negativos.
3) Define os valores percentuais de pares de registros positivos e pares de registros negativos a serem utilizados na etapa de treino.
4) Por fim, executa-se normalmente o processo de deduplicação de registros utilizando PG.

Os experimentos utilizando dados sintéticos mostram que é possível utilizar conjuntos de treino bastante reduzidos para se gerar mais rapidamente as funções de deduplicação. Outra vantagem é de se demandar um tempo de treino consideravelmente inferior, o que é essencial em termos de escalabilidade.

A principal característica da abordagem de [Gonçalves, 2009] é que sua aplicação

é para gerar exemplos de treinamento, ou seja, a partir de um conjunto de dados, seu método cria exemplos de treino mais representativos para posteriormente ser utilizada pelo algorítimo de PG.

A característica comum entre esses trabalhos, é a aplicação das abordagens diretamente no processo de deduplicação de registros. Os resultados são efetivos, porém o custo computacional necessário para o treino é sempre grande. Para exemplificar pode-se observar as abordagens para deduplicação baseado em PG, de [Carvalho et al., 2008a] e [Freitas et al., 2010], que apesar dos bons resultados, a etapa de treino continua com o mesmo custo computacional.

# Capítulo 3

# Conceitos Básicos

Neste capítulo são apresentados alguns conceitos básicos para melhor compreensão da abordagem proposta. Na seção 3.1 são tratados alguns conceitos sobre aprendizagem de máquina, destacando a aprendizagem supervisionada. Na seção 3.2, são descritos os conceitos mais importantes sobre PG. E por fim, na seção 3.3 são apresentados os principais fundamentos sobre técnicas de agrupamento de dados.

## 3.1 Aprendizagem de Máquina

Aprendizado de Máquina (AM) é uma área da Inteligência Artificial responsável pelo desenvolvimento de teorias computacionais focadas na criação do conhecimento artificial, e lida com problemas de aprendizado computacional. Um sistema de aprendizado tem a função de analisar informações e generalizá-las, para a extração de novos conhecimentos. Para isso usa-se um programa de computador para automatizar o aprendizado [Bishop, 2006].

O aprendizado utiliza do princípio da indução (inferência lógica) visando obter conclusões genéricas a partir de um conjunto particular de exemplos. Para a indução derivar conhecimento novo representativo, os exemplos das classes têm que estar bem-definidos e ter uma quantidade suficiente de exemplos, obtendo assim hipóteses úteis para um determinado tipo de problema. Quanto mais exemplos relevantes selecionados para treinamento no indutor, mais bem classificado será o novo conjunto de dados [Bishop, 2006].

De acordo com Bishop [Bishop, 2006], o aprendizado por indução pode ser dividido em:

***Supervisionado:*** É utilizado a figura de um professor externo que apresenta o conhecimento do ambiente por conjuntos de exemplos na forma de entrada e saída desejada.

O algoritmo de AM extrai a representação do conhecimento a partir desses exemplos para classificação. O objetivo é que a representação gerada seja capaz de produzir saídas corretas para novas entradas não apresentadas previamente.
***Não-supervisionado:***Não há a presença de um professor, ou seja, não existem exemplos rotulados. O algoritmo de AM aprende a representar (ou agrupar) as entradas submetidas segundo uma medida de qualidade. Essas técnicas são utilizadas principalmente quando o objetivo for encontrar padrões ou tendências que auxiliem no entendimento dos dados.

O tipo de aprendizado utilizado neste trabalho é o supervisionado. Neste caso, será dado um conjuntos de exemplos rotulados para que o algorítimo de AM procure por padrões podendo usar qualquer informação que seja relevante.

### 3.1.1 Treinamento no Aprendizado Supervisionado

No aprendizado supervisionado é fornecido ao classificador um conjunto de informações organizadas de acordo com a modelagem definida, denominado Conjunto de Treinamento. Através desse conjunto modelado, o classificador é capaz de aprender sobre o problema.

O conjunto de treinamento é construído por uma intervenção externa humana, que define algumas informações do contexto geral como exemplos de treinamento ou testes. Apesar de o conjunto de treinamento ser composto de vários exemplos de treinamento e testes, apenas os exemplos de treinamento são utilizados pelo classificador na aprendizagem do problema [Mitchell, 2007].

Para [Bishop, 2006], deve-se ter cuidado para que o algorítimo não "decore" os dados. Se isso ocorrer, há perda na generalização, ou seja, caso apareça um elemento para o algorítimo identificar, é provável que ele irá errar, pois decorou apenas os dados que foram apresentados a ele anteriormente. Esse problema é chamado *overfitting*.

Uma das técnicas que exigem um conjunto de treino e a etapa de treinamento, é a PG, que será apresentada na seção seguinte.

## 3.2 Programação Genética

A Programação Genética (PG) é uma técnica evolutiva, proposta por [Koza, 1992] inspirada na evolução biológica com base na técnica de Algoritmo Genético (AG), desenvolvida por [Holland, 1975] e aplicada por [Goldberg, 1989]. Ela permite encontrar soluções otimizadas para certas características do problema sendo capaz de realizar busca em grandes espaços, além de permitir trabalhar com características intrínsecas

ao problema. Em [Koza, 1992], com PG é possível criar e manipular software geneticamente, aplicando conceitos herdados da Biologia para gerar programas de computador automaticamente.

Uma das principais características das técnicas evolucionárias é a sua capacidade de tratar problemas com múltiplos objetivos, normalmente modelados como restrições do ambiente durante o processo evolucionário. Essas técnicas também são conhecidas pela capacidade de procurar por soluções em grandes e possivelmente infinitos espaços de busca, nos quais a solução ótima pode ser desconhecida, fornecendo geralmente respostas bem próximas do ótimo ([Koza, 1992]; [Banzhaf et al., 1998]).

Isso ocorre porque o algoritmo PG é capaz de pesquisar, em paralelo, vários pontos do problema apontando inúmeras direções de busca. Para [Koza, 1992] "o fato que o algoritmo de PG opera numa população de indivíduos, em vez de um único ponto no espaço de busca do problema, é um aspecto essencial do algorítimo". Isto pode ser explicado uma vez que a população serve como reserva de o material genético, provavelmente-valioso, que é utilizada para criar novas soluções com novas combinações provavelmente-valiosos de características [Carvalho et al., 2008a].

O principal objetivo de utilizar a PG é encontrar um programa de computador que realize determinada tarefa ou faça determinado calculo. A principal característica que difere PG de outras técnicas evolucionárias (por exemplo, algoritmos genéticos, sistemas evolutivos, sistemas classificadores genéticos) é que ela representa o conceito e à interpretação de um problema como um programa de computador. Programas de computador possuem a flexibilidade necessária para expressar soluções de uma grande variedade de problemas. Esta característica especial permite modelar qualquer outra representação aprendizagem de máquina [Banzhaf et al., 1998].

Normalmente, PG evolui uma população de estruturas de dados com comprimento livre, também chamados de indivíduos, cada um representando uma única solução para um determinado problema.

### 3.2.1 Indivíduo em PG

Segundo [Carvalho et al., 2008a], o indivíduo é um ponto no espaço de otimização. Representa uma solução candidata para o problema e sua codificação tem formato de árvore. O conjunto de indivíduos é denominado população e a geração inicial, na maioria das vezes, é feita de forma aleatória. A quantidade de indivíduos da população é um parâmetro de entrada do PG.

Na abordagem de [Carvalho et al., 2008a] que utiliza PG para deduplicação de registros, por exemplo, os indivíduos são árvores que representam funções aritméticas,

conforme Figura 3.1 abaixo. Nesta representação, cada uma das variáveis (valor numérico) é representada por uma folha na árvore. Os nós internos representam as operações que são aplicadas às folhas, tais como, funções matemáticas simples (por exemplo: $+$, $-$, $\times$, $\div$, $exp$) que manipulam esses valores.

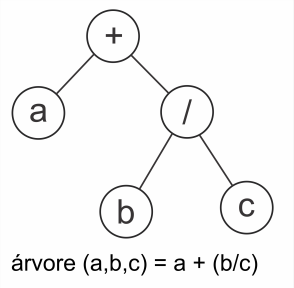


**Figura 3.1.** Exemplo de indivíduo em PG [Carvalho et al., 2008a].

De acordo com [Koza, 1992], o indivíduo em PG deve satisfazer pelo menos duas propriedades, que são:

**1. Suficiência:** garante a convergência do sistema, fazendo com que os conjuntos de nós funções e nós terminais sejam capazes de representar uma solução viável para o problema em questão.

**2. Fechamento:** garante que qualquer função do conjunto de nós funções deve ser capaz de operar com todos os valores recebidos como entrada. Isso garante que sejam geradas árvores sintaticamente viáveis.

O fato do indivíduo na PG ser representado na forma de árvore possibilita a esse indivíduo conter não somente valores de variáveis, mas também funções. O termo que melhor se aplica a esse conjunto é conjunto não terminal, uma vez que o conjunto de nós terminais pode conter funções que não recebem parâmetros. O termo conjunto de funções será adotado a partir de agora, por ser o mais utilizado na literatura, devido à grande influência exercida pelo livro de [Koza, 1992]. Assim, é importante que as árvores sejam manipuladas por operações genéticas capazes de evitar situações que poderiam afetar a integridade da função global.

Em PG, existem várias maneiras de gerar a população inicial de indivíduos que vai participar no processo evolutivo. A maneira mais comum é a criação de árvores aleatórios, utilizando as funções e os terminais disponíveis. Algumas abordagens também fazem uso de a entrada do usuário para sugerir a criação da população inicial. Isto é conseguido através da utilização de uma árvore (possível solução do problema) para a criação de variantes, por exemplo, através da aplicação de mutações ou cruzando-os com outras árvores criadas ao acaso [Carvalho et al., 2008a].

De acordo com [Banzhaf et al., 1998], a partir do momento em que a representação de árvore foi escolhida para representar os indivíduos, é importante que após a aplicação de cada operação genética, as árvores resultantes ainda sejam árvores válidas. Por exemplo, um nó folha nunca é substituído por um nó interno e vice-versa.

Pode se dizer que os indivíduos são as possíveis soluções de um problema e serão submetidos a um processo evolutivo e logo avaliados por uma função de avaliação (descritas na seção 3.2.1.1) especifica do problema, ou seja, são manipulados e modificados por operações genéticas (descritas na seção 3.2.2) como reprodução, cruzamento (*crossover*) e mutação, em um processo iterativo que tenta gerar indivíduos cada vez melhores e mais aptos para cada geração subsequente.

#### 3.2.1.1 Função de *Fitness* (aptidão)

A função de avaliação (ou aptidão), mais conhecida na literatura da área como função de *fitness*, avalia a qualidade de um indivíduo para a resolução de determinado problema. Ela é o componente da PG que deve expressar o conhecimento acerca do problema para tornar-se eficiente/eficaz a tarefa de garantir que os indivíduos com a melhor satisfatibilidade para a solução do problema sejam identificados. De nada adiantará possuir uma PG bem estruturada se a função *fitness* não estiver de acordo com o escopo do problema [Carvalho et al., 2008a].

A função *fitness* pode ser definida de duas maneiras: quanto maior o valor retornado por ela, melhor será a avaliação do indivíduo, ou o contrário, onde quanto menor o valor retornado pela função, melhor será a avaliação do indivíduo. Outro ponto importante na função é de que cada indivíduo (programa) executa várias instâncias diferentes de entradas, e provavelmente consumirá mais tempo durante o cálculo do seu *fitness* [Linden, 2012].

### 3.2.2 Operadores Genéticos

No processo evolutivo, os indivíduos são manipulados e modificados pelas seguintes operações genéticas: reprodução, cruzamento (*crossover*) e mutação. O objetivo é criar um processo interativo onde são gerados indivíduos mais aptos para as próximas gerações. Esses três operadores genéticos serão descritos nas seções seguintes.

#### 3.2.2.1 Reprodução

A Reprodução tem como base a seleção natural Darwiniana. Ela ocorre em duas etapas: 1) Um individuo da população é escolhida através de algum método de seleção baseado

em sua *fitness*;
2) O individuo selecionado é inserido, sem qualquer alteração, da população atual para a nova população [Koza, 1992].

Isso consiste em copiar os indivíduos sem realizar qualquer tipo de modificação em suas estruturas. Geralmente, esta operação é utilizada para implementar uma estratégia elitista, mantendo o código genético dos indivíduos mais aptos inalterados no decorrer das gerações.

Dessa forma, se um bom indivíduo é encontrado nas gerações iniciais, dificilmente ele será perdido durante o processo evolucionário, após diversas aplicações de operações genéticas.

#### 3.2.2.2 Crossover

A operação de *Crossover* ou cruzamento, tem como base a troca de conteúdo genético entre dois ou mais indivíduos pais, resultando em dois indivíduos filhos. Intuitivamente, se dois indivíduos são pelo menos um pouco efetivos na resolução de um determinado problema, então alguma de suas partes provavelmente possui algum mérito.

Ao realizar a recombinação dos fragmentos de alguns bons indivíduos, espera-se que sejam gerados indivíduos ainda melhores que seus pais, capazes de resolver o problema com maior êxito. Esta operação pode ser exemplificada conforme Figura 3.2 [Carvalho et al., 2008a].

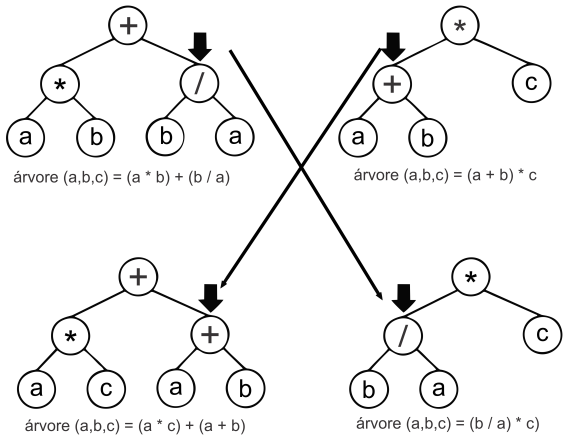


**Figura 3.2.** Cruzamento em PG [Carvalho et al., 2008a].

Esta operação pode ser descrita em quatro passos [Carvalho et al., 2008a]:
**Passo 1.** Seleciona-se dois indivíduos (árvores pais) de acordo com alguma política de pareamento.

**Passo 2.** Escolhe-se aleatoriamente um fragmento de cada indivíduo (sub-árvore).
**Passo 3.** Permuta-se os dois fragmentos escolhidos.
**Passo 4.**. Reinicia-se o processo evolucionário com os indivíduos resultantes do cruzamento (árvores filhas).

Na operação de cruzamento, todos os nós das árvores pais apresentam a mesma probabilidade de serem escolhidos, visando a manutenção da diversidade dos indivíduos dentro da população. Uma característica bastante significativa desta operação é a sua capacidade de alterar os tamanhos dos indivíduos durante a execução do algoritmo.

#### 3.2.2.3 Mutação

De acordo com [Carvalho et al., 2008a], o operador de mutação não é simples de ser implementado. É necessário ter certeza de que a árvore do indivíduo se manterá válida após a operação de mutação. Esse operador realiza a troca de um nó (terminal ou não) por outro nó (terminal ou uma sub-árvore) gerado. Assim como no cruzamento, os indivíduos podem aumentar ou diminuir de tamanho. A probabilidade do indivíduo sofrer mutação, é um parâmetro de entrada do algorítimo de PG.

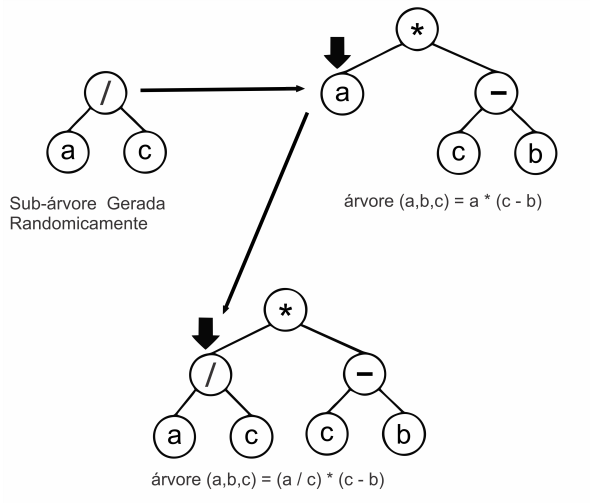


**Figura 3.3.** Mutação em PG [Carvalho et al., 2008a].

A Figura 3.3, exemplifica o processo de mutação, que visa a manutenção de um nível mínimo de diversidade dos indivíduos na população, ajudando a PG a obter boas soluções mais rapidamente.

O processo de mutação é descrito em quatro passos [Carvalho et al., 2008a]:

**Passo 1.** Seleciona-se um indivíduo da população atual[1].
**Passo 2.** Seleciona-se aleatoriamente um fragmento desse indivíduo.
**Passo 3.** Cria-se aleatoriamente uma árvore de mutação.
**Passo 4.**. Gera-se um novo indivíduo através da substituição do fragmento do indivíduo selecionado pela árvore de mutação criada no passo anterior.

O cruzamento e a mutação, por serem responsáveis pela modificação de soluções candidatas em novas soluções candidatas para o problema, são denominadas operações de transformação [Banzhaf et al., 1998].

### 3.2.3 Processo Evolucionário

Existem duas maneiras de realizar o processo evolutivo em PG: através da aplicação um algoritmo de estado estacionário que não considera gerações distintas, ou um geracional que considera gerações distintas [Banzhaf et al., 1998].

Um algoritmo de estado estacionário copia a próxima geração de indivíduos para a mesma população da qual os pais foram selecionados anteriormente. Quando as operações genéticos sobre os pais são concluídas, a nova prole toma o lugar dos membros da geração anterior dentro dessa população. Em fim, os novos indivíduos da população são trocados com os antigos membros da mesma população. Este processo de criação da nova população continua até que não haja indivíduos da geração anterior ou outro critério de término é estabelecido, tal como o tempo de processamento. Assim, neste caso, o processo evolutivo flui num funcionamento contínuo [Banzhaf et al., 1998].

Por outro lado, um algoritmo geracional compreende de forma bem definida e distinta os ciclos de geração. Os passos que compreendem este tipo de algoritmo são:
1) Inicializar a população (com os indivíduos aleatórios ou fornecidos pelo usuário).
2) Avalie todos os indivíduos na população atual, atribuindo uma classificação numérica ou valor de *fitness* para cada um;
3) Se o critério de término for cumprido, executar a última etapa. Caso contrário, continue.
4) Reproduzir os melhores $n$ indivíduos para a próxima geração na população.
5) Selecione $m$ indivíduos que irão compor a próxima geração com os melhores pais.
6) Aplique as operações genéticas a todos os indivíduos selecionados. Seus descendentes irão compor a próxima população. Substituir a geração existente pela população gerada e voltar para a Etapa 2.
7) Apresentar o melhor indivíduo(s) na população como a saída do processo evolutivo.

[1]Cada árvore resultante da etapa de cruzamento possui chances iguais de sofrer mutação.

A avaliação no Passo 2 é feita através da atribuição de um valor ao indivíduo, que mede quão adequado aquele individuo é para o problema proposto. Em [Carvalho et al., 2008a], os indivíduos são avaliados em quão bem eles aprendem para prever boas respostas a um determinado problema, usando o conjunto de funções e terminais disponíveis. O valor resultante é o *fitness* bruto e as funções de avaliação são as funções de *fitness*.

O valor de *fitness* pode ser visto como uma "direção"ou "caminho"de um solução sugerida pela PG para o processo evolutivo. Assim, utilizando este valor de *fitness*, no Passo 5, é possível selecionar quais os indivíduos devem seguir para a próxima geração.

A etapa de seleção aplica um critério de escolha dos indivíduos que devem estar na próxima geração. Após a avaliação, cada solução tem um valor de *fitness* que mede o quão bom ou ruim ela é para o problema dado. Utilizando este valor, é possível decidir se um indivíduo deve estar na próxima geração.

Cada execução deste laço representa uma nova geração de programas. Tradicionalmente, o critério de término é estabelecido como sendo encontrar uma solução satisfatória ou atingir um número máximo de gerações.

Para melhor compreensão, esse algoritmo é apresentado detalhadamente no fluxograma da Figura 3.4.

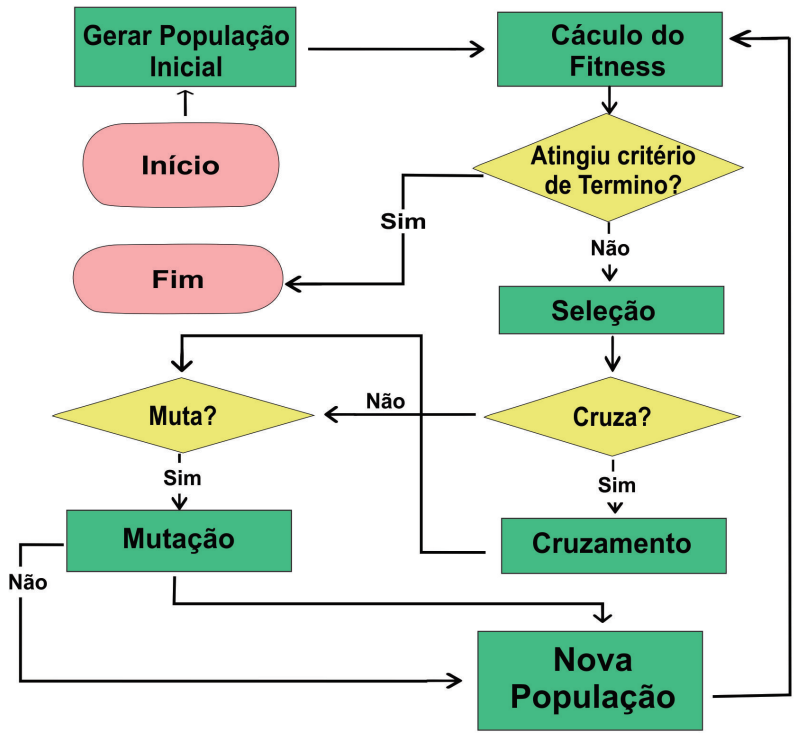


**Figura 3.4.** Fluxograma do funcionamento básico de PG [Carvalho et al., 2008a].

A execução iterativa desses procedimentos permite a emergência de boas soluções a cada geração. A PG assume que existe uma função no espaço de soluções do problema que é capaz de separar os indivíduos bons dos indivíduos ruins utilizando a função de *fitness*. O processo evolucionário tenta por meio das suas manipulações fazer regressão até essa função.

### 3.2.4 Etapa de Treinamento em PG

A etapa de treinamento busca identificar as características relevantes de boas soluções (que são dadas como exemplos). Na maioria das vezes, isso é realizado com a utilização de: exemplos positivos, ou seja, soluções válidas ou aceitáveis; exemplos negativos, que são soluções inválidos ou inaceitáveis, ou os dois ao mesmo tempo.

Para [Carvalho et al., 2008a], ao usar PG (e outras técnicas de aprendizagem de máquina), há dois pontos importantes que devem ser observados para garantir soluções generalizáveis, que são:

**Amostragem:** é um fator de impacto, pois se as técnicas de amostragem forem ruim levarão a um subconjunto da população com características diferentes em relação os encontrados na base de dados. Por esta razão, a amostragem tem de ser realizada com cuidado, a fim de assegurar conjuntos de treino e de avaliação válidas. Caso contrário, as técnicas de aprendizagem não fornecerão soluções úteis.

***Overfitting:*** ocorre quando uma proposta de solução funciona com êxito no conjunto de treino, mas o seu comportamento no conjunto de avaliação é muito diferente, geralmente fornecem resultados pobres. Assim, que ao invés de melhorar o desempenho, ele começa a piorar.

O tamanho do conjunto de treinamento é um fator importante sendo que quanto menor o conjunto de treinamento (ou seja, não representativa), a menos de confiança serão as soluções obtidas a partir deste conjunto.

## 3.3 Agrupamento de Dados

Agrupamento, é o nome dado para o grupo de técnicas computacionais cujo propósito consiste em separar objetos em grupos, baseando-se nas características que estes objetos possuem. A ideia básica consiste em colocar no mesmo grupo objetos que sejam similares de acordo com algum critério pré-determinado [Bishop, 2006].

Algoritmos de agrupamento têm por objetivo particionar um conjunto de dados em *clusters* de tal forma que indivíduos dentro de um mesmo *cluster* tenham um alto grau de similaridade, enquanto indivíduos pertencentes a diferentes *clusters* tenham

alto grau de dissimilaridade. O critério da dissimilaridade baseia-se em uma função que recebe dois objetos e retorna a distância entre eles.

Nessa técnica, o procedimento inicia com o cálculo das distâncias entre os objetos estudados dentro do espaço multiplano constituído por eixos de todas as medidas realizadas (variáveis), sendo, a seguir, os objetos agrupados conforme a proximidade entre eles. Na sequência, efetuam-se os agrupamentos por proximidade geométrica, o que permite o reconhecimento dos passos de agrupamento para a correta identificação de grupos dentro do universo dos objetos estudados [Bishop, 2006].

[Ankerst et al., 1999] destacam que o objetivo de descobrir *clusters* é de encontrar a partição de um banco de dados de registros tal que os registros que tem características semelhantes são agrupados juntos. Isso, então, permite que as características de cada grupo possam ser descritas.

A técnica de agrupamento tem por objetivo agrupar automaticamente os $n$ casos da base de dados em $k$ grupos, geralmente disjuntos. Ela pode ser usada como uma etapa de pré-processamento para outros algoritmos, tais como caracterização e classificação. Quando objetivo é reduzir o número de objetos, para um número de subgrupos característicos, essa técnica é ideal [Bishop, 2006].

### 3.3.1 Métodos de Agrupamento

Análise de agrupamento ou análise de *cluster* é a organização de uma coleção de padrões (usualmente representado como vetores de medidas ou um ponto em um espaço multidimensional) em *clusters* tomando como critério a similaridade. A literatura atual descreve várias abordagens para agrupamento e sua utilização depende tanto dos tipos de dados disponíveis quanto da aplicação que se deseja realizar [Bishop, 2006].

Os métodos mais gerais de agrupamento são apresentados por [Jain et al., 1999], que são: Hierárquico, Particionado, Baseados em Densidade, Baseados em Grades e Baseados em Modelos. Os métodos mais utilizados são o hierárquico e de particionamento. Esses dois métodos serão descritos resumidamente nas subseções seguintes.

#### 3.3.1.1 Métodos Hierárquicos

Os algoritmos baseados nos métodos hierárquicos organizam um conjunto de dados em uma estrutura hierárquica de acordo com a proximidade entre os indivíduos. Os resultados desse algorítimo são normalmente representandos por árvore binária ou um dendograma.

Conforme [Jain et al., 1999] os algorítimos hierárquicos são divididos em aglomerativos ou divisivos. Algorítimos aglomerativos começam com cada indivíduo em um

conjunto de dados sendo representado por um único *cluster*. Após isso, ocorre uma série de combinações entre os *clusters* iniciais onde o resultado será que todos indivíduos estarão em um mesmo *cluster*. Algorítimos divisivos, iniciam com um conjunto de dados em um só *cluster* e um procedimento que divide os *clusters* vai ocorrendo sucessivamente até que cada indivíduo seja um único *cluster*.

Os passos genéricos de um algorítimo hierárquico aglomerativo são mostrados conforme pseudocódigo abaixo [Jain et al., 1999]:

**Algoritmo 1:** Algoritmo hierárquico aglomerativo

: Comece com $N$ *clusters* cada um com um único indivíduo.

: Calcula a matriz de proximidade para os $N$ *clusters*.

: Procure a distância mínima:
$d(C_i, C_j) = \min\ d(C_m, C_l)$
$l \leq m, l \leq N$
$m \neq l$
onde $d(*,*)$ é calculada a partir da matriz de proximidade. Esta distância é usada para formar um novo *clusters* a partir dos *clusters* $C_i$ e $C_j$.

: Atualize a matriz de proximidade, calculando a distância entre os novos *clusters* e os outros.

: Repita os passos 3 e 4 até que todos os indivíduos estejam no mesmo *cluste*r.

Existem muitos algorítimos aglomerativos, entre os mais simples e populares estão os métodos de ligação simples e ligação completa. Na ligação simples, a distância entre dois *clusters* é determinada pelos dois indivíduos, cada um de um cluster diferente, mais próximos um do outro. Ao contrário, a ligação completa usa como distância entre dois *clusters* aquela entre os indivíduos nos diferentes *clusters* que estão mais afastados um do outro [Jain et al., 1999].

De forma geral, o método Hierárquico cria uma decomposição hierárquica da base de dados. A decomposição hierárquica é representada por uma árvore que iterativamente divide a base de dados em subconjuntos menores até que cada um desses subconjunto consista de somente um objeto [Jain et al., 1999].

#### 3.3.1.2 Métodos de Particionamento

Ao contrário de métodos hierárquicos, métodos de partição associam um conjunto de indivíduos a $K$ grupos sem criar uma estrutura hierárquica. Esses algorítimos geralmente são baseados em algoritmos gulosos cujo princípio é encontrar uma solução aproximada para tomar a melhor decisão no momento, esperando que isto conduza a melhor decisão global [Jain et al., 1999].

Os algorítimos de particionamento dividem a base de dados em $k$ grupos, onde o número $k$ é dado pelo usuário. Os objetos são divididos entre os $k$ *clusters* de acordo com a medida de afinidade adotada, de modo que cada objeto fique no cluster que forneça o menor valor de distância entre o objeto e o centro do mesmo. Então é utilizada uma estratégia iterativa de controle para determinar que objetos devem mudar de cluster de forma que otimizemos a função objetivo usada.

Quando comparado com o método hierárquico, o método por particionamento é mais rápido porque não é necessário calcular e armazenar, durante o processamento, a matriz de similaridade. Em geral, os métodos por particionamento diferem entre si pela maneira que constituem a melhor partição.

Os métodos de particionamento mais utilizados são: baseados em um ponto central, média dos atributos dos objetos - k-médias (*k-means*) [Zhang et al., 2001]; ou em um objeto representativo para o *cluster k-medoids* [Zaufman, 1990].

O k-médias é voltado para aplicações em que todas as variáveis são quantitativas e as dissimilaridades entre elas podem ser medidas em um espaço Euclidiano. K-medoids é uma generalização do k-médias no sentido que outras funções de distâncias, além da distância Euclidiana, podem ser utilizadas para medir as dissimilaridades entre os elementos [Zhang et al., 2001].

A idéia por trás do algoritmo k-médias é escolher $k$ objetos (aleatoriamente ou com alguma heurística) que servirão como base para cada grupo (denominado centróide), os outros objetos ficam associados ao centroide que se encontrar mais próximo. Durante cada passo os centroides precisam ser recalculados em relação aos objetos de seu próprio grupo, em seguida esses objetos são realocados para o centroide que estiver mais próximo. Repete-se este procedimento até que o nível de convergência seja alcançado satisfatoriamente.

O algoritmo *K-médias* pode ser descrito conforme pseudocódigo abaixo [Zhang et al., 2001]:

**Algoritmo 2:** Algoritmo K-médias básico

Selecione K pontos como centroides iniciais

**repita**

  Forme K grupos atribuindo cada ponto ao seu centroide mais próximo.

  Recalcule o centroide de cada grupo

**até *que*** *Nenhum centróide mude de lugar*;

O algorítimo *K-means* é bem simples e pode ser facilmente implementado para resolver muitos problemas práticos. Ele é adequado para *clusters* compactos e hiperesféricos. compactos e hiperesféricos.

O algorítimo K-medoid foi introduzido por Kaufman e Rousseeuw

([Zaufman, 1990]) e não é sensível a anômalos (*outliers*) como o *K-médias*. Neste método de agrupamento, cada grupo é representado pelo objeto mais centralizado conhecido como medóide. A forma de agrupamento que o K-medoid utiliza é de acordo com o algoritmo PAM (*Particioning Around Medoids*), conforme descrito abaixo:

**Algoritmo 3:** Algoritmo K-medoid básico

1. Escolher aleatoriamente K objetos como medóides iniciais;
2. Atribuir, para cada um dos objetos restantes, um grupo que tenha o medóide mais próxima;
3. Em um grupo, selecionar aleatoriamente os objetos não medóide, que são chamados $O(nonmedoid)$
4. Computar o custo de trocar o medóide com $O(nonmedoid)$
5. Repetir a partir do passo 2 até não haver mais mudanças.

O K-medoid define um protótipo em termos de um medóide (*medoid*), que trata-se de um ponto mais representativo em relação a um grupo de pontos e pode ser aplicado para uma grande quantidade de dados, já que requer apenas uma medida para proximidade em relação a um par de objetos.

# Capítulo 4

# Deduplicação de Registros com PG

Neste capítulo é detalhada a utilização da técnica de PG no contexto do problema de deduplicação de registros, sendo descrita a abordagem proposta por [Carvalho et al., 2008a] e que serve como arcabouço para o trabalho desenvolvido nesta dissertação.

## 4.1 Deduplicação de Registros

Deduplicação de registros é a tarefa de identificar, em um repositório de dados, registros similares que se referem à mesma entidade do mundo real ou objeto, apesar das palavras com erro de ortografia, erros de digitação, diferente estilos de escrita ou representações de esquema, mesmo diferentes ou tipos de dados [Carvalho et al., 2008a].

É interessante notar o uso da palavra similar ao invés da palavra igual, pois, dada uma abordagem determinística de comparação de registros que utilize um identificador unívoco ou busque por registros idênticos, as igualdades são facilmente detectáveis, porém, o mesmo não acontece para a identificação de registros similares [Carvalho et al., 2008a].

Um dos primeiros trabalhos visando identificar registros duplicados foi proposto por [Newcombe et al., 1959]. Utilizando uma abordagem probabilística, a proposta era fazer uma avaliação fonética dos nomes e/ou sobrenomes, calculando um peso para cada atributo, comparando-os para definir se os registros seriam classificados como similares ou não similares.

Com base na visão de [Newcombe et al., 1959], os pesquisadores Ivan Fellegi e Alan Sunter ([Fellegi, 1969]), propuseram um processo de identificação de duplicidades fortemente sedimentado. Basicamente, o método consiste em comparar os dados fazendo a classificação dos registros e gerando como resultado três conjuntos de dados:

A1 que contém os registros considerados similares, A2 com pares possivelmente similares ou duvidosos e A3 com os registros não similares. Até os dias atuais, a visão de [Newcombe et al., 1959] é utilizada como base para diversos outros métodos que visam realizar a deduplicação de registros, como por exemplo [Carvalho et al., 2008a] que propôs uma abordagem para deduplicação de registros utilizando PG. Essa abordagem será descrita nas seções seguintes.

## 4.2 Modelagem para Problema de Deduplicação de Registros com PG

Ao utilizar PG (ou mesmo alguma outra técnica evolutiva) para resolver um problema, existem alguns requisitos básicos que devem ser cumpridos, que são baseados na estrutura de dados usada para representar a solução. Na abordagem de [Carvalho et al., 2008a], os autores escolheram uma representação de PG baseada em árvore para a função de deduplicação, uma vez que é uma representação natural para esse tipo de função (isto é, combinação de valores numéricos).

Partindo desse pressuposto, esses requisitos são os seguintes [Banzhaf et al., 1998]:

1. Todas as soluções possíveis para o problema devem ser representadas por uma árvore, não importa o seu tamanho.
2. As operações evolutivas aplicadas sobre cada árvore devem, no final, resultar em uma árvore válida.
3. Cada indivíduo deve ser avaliado automaticamente.

Para o requisito 1, é necessário levar em consideração o tipo de solução que pretende encontrar. Para realizar a deduplicação de registros, são utilizadas funções que combinam evidências de similaridade entre os atributos do objeto A e o objeto B. Cada evidência $E$ é formada por um par $<$*atributo, função de similaridade*$>$ que representa o uso de uma função de similaridade específica sobre valores de um determinado atributo do repositório de dados. Assim, a deduplicação baseada em programação genética pode ser entendida conforme [Carvalho et al., 2008a], partindo do seguinte exemplo:

Para deduplicar a tabela de um banco de dados relacional com os atributos *nome, sobrenome, idade e endereço*, utilizando a função de similaridade Jaro-Winkler (JW) proposta por [Winkler, 1999], têm se a seguinte lista de evidências:

$E_1 <$*nome,* JW$>$, $E_2 <$*sobrenome,* JW$>$, $E_3 <$*idade,* JW$>$ e $E_4 <$*endereço,* JW$>$.

Para este exemplo, uma função simples *($F_s$)* poderia ser uma combinação linear da forma

$$F_s(E_1, E_2, E_3, E_4) = E_1 + E_2 + E_3 + E_4,$$

enquanto uma função mais complexa *($F_c$)* poderia ser da forma

$$F_c(E_1, E_2, E_3, E_4) = E_1 \times (\frac{E_2}{E_3^{E_4}}).$$

Para modelar as funções em formato de árvore, cada evidência é representada por uma folha, através de valores reais normalizados entre 0,0 e 1,0. A folha pode também ser um número aleatório entre 1,0 e 9,0, que é escolhido no momento em que cada árvore é gerada. Tais folhas (números aleatórios) são usados para permitir que o processo evolutivo para encontrar os pesos mais adequados para cada prova, quando necessário. Os nós internos representam operações aritméticas (por exemplo: $+$, $-$, $\times$, $\div$, *exp*) que manipulam os valores da folha.

Para aplicar o Requisito 2, as árvores são manipulados por operações atômicas nas sub-árvore [Banzhaf et al., 1998], para evitar situações que possam afetar a integridade da função global. Não pode ser nem um caso em que o valor de um nó de folha é substituído pelo valor de um nó interno, nem um em que o valor de um nó interno é substituído pelo valor de um nó de folha [Koza, 1992].

Enquanto ocorre o processo evolucionário, diversas operações genéticas manipulam e modificam os indivíduos, buscando sempre gerar os indivíduos com melhor aptidão para as próximas gerações. A avaliação das árvores geradas é realizada de forma automática, ou seja, cada possível solução para o problema é testada em repositórios de dados onde as réplicas já foram previamente identificadas.

No Requisito 3, todas as árvores geradas durante um processo evolutivo de PG são testadas em repositórios de dados pré-avaliados onde as réplicas foram previamente identificados. Isto torna viável para executar automaticamente todo o processo, uma vez que é possível avaliar o modo como as árvores realizam a tarefa de reconhecimento dos pares de registo que são réplicas.

Após os dados serem manipulados, são extraídas as instâncias de evidências que formarão as entradas das funções. A saída da função está condicionada ao resultado da operação codificada em cada árvore. O valor de saída é comparado com um limiar de identificação de réplicas da seguinte forma: se o valor for superior ao limiar, os registros são considerados réplicas (*link*); caso contrário, os registros são considerados distintos (*non-link*).

Essa abordagem de classificação obedece as propriedades de transitividade das réplicas, de forma que, se um registro $A$ for réplica de um registro $B$ e $B$ for réplica de um registro $C$, então $A$ será réplica de $C$, e assim sucessivamente.

Em [Carvalho et al., 2008a], os autores apresentam uma visão geral de como ocorre a deduplicação de registros utilizando a abordagem baseada em PG, conforme Figura 4.1:

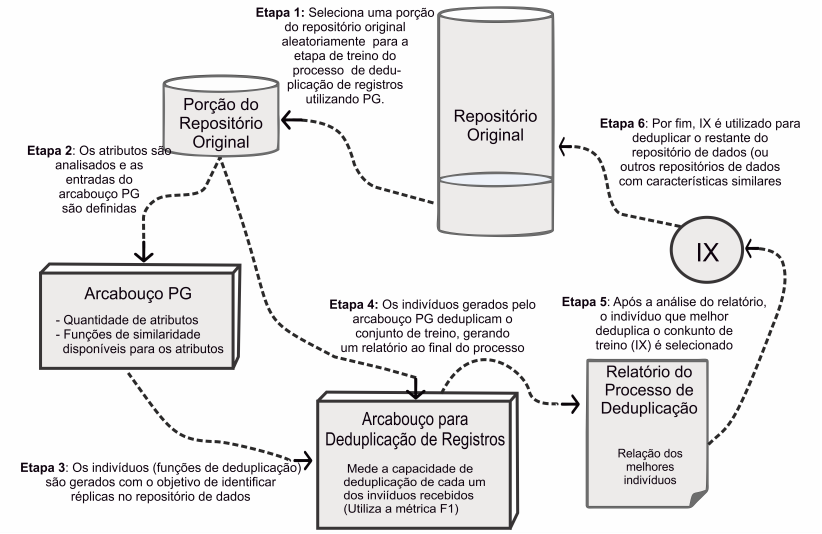


**Figura 4.1.** Esquema de Deduplicação utilizando PG [Gonçalves, 2009].

Uma descrição detalhada de cada etapa exposta na Figura 4.1 é apresentada a seguir [Gonçalves, 2009]:

**Etapa 1:** Seleciona aleatoriamente uma parte do repositório de dados a ser deduplicado, e utiliza na etapa de treino. No caso dos experimentos de [Carvalho et al., 2008a], o repositório foi dividido quatro arquivos com a mesma quantidade de registros e um desses arquivos foi utilizado para treino da PG. O restante dos arquivos foram utilizados para a avaliação (teste) dos indivíduos gerados.

**Etapa 2:** Os atributos do repositório de dados são analisados e as entradas do arcabouço PG são definidas, ou seja, verifica-se o tipo de cada um dos atributos para então selecionar as funções de similaridade mais adequadas para cada um deles. Desse modo, o arcabouço PG mantém uma lista com os atributos e as respectivas funções de similaridades utilizadas para a criação das evidências.

**Etapa 3:** Ao final de cada geração do processo evolucionário, são selecionados indivíduos (possíveis soluções para o problema) com o objetivo de se identificar registros réplicas em um determinado repositório de dados.

**Etapa 4:** No arcabouço para deduplicação de registros, os indivíduos gerados são utilizados para identificar réplicas na porção do repositório de dados utilizada para teste, gerando um relatório ao término do processo evolucionário. Neste relatório, é apresentada a relação dos melhores indivíduos de cada geração e os seus respectivos valores de aptidão, medidos pela métrica F1.

**Etapa 5:** Após uma análise do relatório do processo de deduplicação, é realizado o passo seguinte onde o melhor indivíduo obtido, ou seja, a função que melhor conseguiu deduplicar o conjunto de dados de treino.

**Etapa 6:** O melhor indivíduo obtido é então utilizado para deduplicar o restante do repositório de dados, podendo também ser utilizado para a deduplicação de outros repositórios com características semelhantes.

### 4.2.1 Definição sobre Pares de Registros

Um registro é constituído por conjunto de campos valorados, ou seja, são os locais onde os itens individuais de informações são armazenados. Cada registro consiste em um ou mais campos. Os campos correspondem às colunas da tabela. Geralmente, os registros de um arquivo possuem um formato padrão, definido pela sequência, tipo e tamanho dos campos que o compõem.

Um exemplo de como os registros são dispostos em um repositório é apresentado na Figura 4.2:

| CodCli | Nome do Cliente | Contato | Endereço | Estado |
|---|---|---|---|---|
| MARAT | Maria Raimunda Tan | Maria Ray Tan | Rua 22, nº 35 | TO |
| ANMOP | Ana Morata Pena | Ana Morata | Av. del liro, 1765 | AM |
| MARTA | Maria Raimunda Tan | Maria Raimunda | Rua 22, num 35 | TO |
| SIMAN | Simão Antero | Mara Jessica | Tv. Montrev, 323 | RJ |
| NISAN | Nilzo Santos | Jorge Leite | Conj. Homby, 987 | SP |
| MARLE | Marjory Leal | Marjory | Av. Castell, 111 | AM |
| AMSSO | Adriana Santos Sé | Adriana Sé | Tv 1. n987 | RJ |
| LULEL | Luiz Léo Lima | Luiz Lima | Rua 8. 456 | TO |
| ROTIL | Ronaldo Tito Lopes | Ronaldo L. | Tv 1. n987 | RJ |

**Figura 4.2.** Exemplo de registros e seus atributos.

Para realizar a deduplicação de registros com a abordagem de PG, eles são agrupados em pares para serem comparados por funções de similaridade previamente definidas pelo usuário. O objetivo é que os resultados exibam quais registros do repositório são realmente réplicas nos exemplos de treino.

De acordo com [Gonçalves, 2009], ao falar sobre exemplos de treino, é necessário introduzir os conceitos de pares de registros positivos e pares de registros negativos. Um

par de registros positivo é formado por dois registros que fazem referência ao mesmo objeto ou entidade do mundo real. Por outro lado, um par de registros negativo é constituído por dois registros que não fazem referência ao mesmo objeto do mundo real, não sendo apontados como réplicas um do outro.

Esses dois tipos de exemplo de treino auxiliam o método de deduplicação de registros utilizando PG a compreender o que é e o que não é réplica dentro de um repositório[Gonçalves, 2009].

### 4.2.2 Função de *Fitness* na Deduplicação

Como explicado anteriormente, cada árvore representa uma função que é usada para encontrar réplicas em um repositório. Esta função é aplicada a todas as comparações entre os registros que ocorrem durante a deduplicação.

Ao finalizar o processo de comparação entre todos os pares de registros selecionados, ocorre a contabilização da quantidade de réplicas que foram identificadas corretamente e incorretamente. Essa informação é importante para calcular a função de aptidão, que é responsável pela avaliação dos indivíduos gerados durante todo o processo evolucionário. Essa métrica, chamada *F1*, combina harmonicamente as tradicionais métricas de precisão e revocação utilizadas em avaliações de sistemas de recuperação de informação [Baeza-Yates, 2011] e [Bilenko, 2003].

Essa combinação é definida pelas seguinte fórmulas:

$$Precisão = \frac{QuantidadeDeParesDeRéplicasIdentificadasCorretamente}{QuantidadeDeParesDeRéplicasIdentificados}$$

$$Revocação = \frac{QuantidadeDeParesDeRéplicasIdentificadasCorretamente}{QuantidadeDeParesDeRéplicasExistentes}$$

$$F1 = \frac{2 \times Precisão \times Revocação}{Precisão + Revocação}$$

A precisão é responsável por mensurar a proporção de réplicas identificadas corretamente dentre todas as identificações realizadas, ou seja, de todos os pares de registros que foram identificados como réplicas pela função de deduplicação, quantos pares realmente são réplicas. Já a revocação é utilizada para calcular a proporção de réplicas identificadas corretamente dentre todas as identificações que deveriam ter sido feitas, ou seja, de todos os pares de registros que deveriam ter sido identificados como réplicas, quantos foram devidamente identificados [Gonçalves, 2009].

Uma vez que a precisão e a revocação são métricas relacionadas, capazes de capturar diferentes aspectos da identificação de réplicas no contexto de deduplicação de registros, decidiu-se pela utilização de uma única métrica capaz de combinar precisão

e revocação [Baeza-Yates, 2011]. Dessa forma, a métrica F1 foi utilizada para representar, através de um único valor (entre 0 e 1), o quão bem um indivíduo consegue identificar réplicas em um repositório de dados. É importante ressaltar que a métrica F1 assume valores elevados apenas quando os valores de precisão e revocação também forem elevados [Gonçalves, 2009].

### 4.2.3 Complexidade da Etapa de Treinamento

Conforme [Christen, 2007], para se identificar réplicas em um repositório, cada registro deve ser comparado com todos os demais. Logo, se determinado repositório contém $n$ registros, devem ser realizadas $O(\frac{n\times(n-1)}{2})$ comparações entre dois registros. O divisor é igual a dois pois, na realidade, apenas metade das comparações é realizada, já que dois registros nunca são comparados mais de uma vez.

A maioria das comparações corresponderão a não-réplicas (pares de registros negativos), uma vez que o número máximo de pares duplicados é geralmente menor que a quantidade de registros no repositório.

A complexidade de tempo da etapa de treinamento, com base na modelagem proposta por [Carvalho et al., 2008a] e serve como arcabouço para nossa proposta, é $O(N_g \times N_i) \times T_e$, onde $N_g$ é o número de gerações de evolução, $N_i$ é o número de indivíduos na piscina população e $T_e$ é complexidade da avaliação da função de *fitness* de um indivíduo.

Nessa abordagem, a complexidade da função de *fitness* de um indivíduo é a complexidade de um único processo de deduplicação dado por $O(N_t)$, onde $N_t$ é o número de pares de formação. Esse processo exige uma comparação de cada registro com todos os outros registos disponíveis no repositório a ser processado. Assim, a complexidade da formação é dada por $O(N_g \times N_i \times N_t)$.

Este é o pior cenário possível para a etapa de treinamento pois dependendo da quantidade de registros e as características das amostras de treino, essa etapa pode demandar muito tempo e custo computacional . Esse é grande problema que dificulta a adoção dessa técnica.

Para exemplificar, na Figura 4.3 é apresentado um cenário de um repositório contendo registros positivos e negativos que serão comparados na etapa de treinamento. De modo geral, será necessário realizar a comparação entre todos esses registros para encontrar as possíveis duplicatas existentes.

| CodCli | Nome do Cliente | Contato | Endereço | Estado |
|---|---|---|---|---|
| MARAT | Maria Raimunda Tan | Maria Ray Tan | Rua 22, nº 35 | TO |
| ANMOP | Ana Morata Pena | Ana Morata | Av. del liro, 1765 | AM |
| MARTA | Maria Raimunda Tan | Maria Raimunda | Rua 22, num 35 | TO |
| SIMAN | Simão Antero | Mara Jessica | Tv. Montrev, 323 | RJ |
| NISAN | Nilzo Santos | Jorge Leite | Conj. Homby, 987 | SP |
| MARLE | Marjory Leal | Marjory | Av. Castell, 111 | AM |
| NISAN | Nilzo Santos | Jorge Leite | Tv 1. n987 | RJ |
| LULEL | Luiz Léo Lima | Luiz Lima | Rua 8. 456 | TO |
| ROTIL | Ronaldo Tito Lopes | Ronaldo L. | Tv 1. n987 | RJ |

**Figura 4.3.** Exemplo de pares de registros em um repositório.

A Figura 4.3, mostra um cenário observado comumente em repositórios reais e sintéticos. Trata-se dos registros negativos que aparecem em maior quantidade em relação a positivos. Em [Gonçalves, 2009] é descrito que em cenários reais, a proporção de registros positivos em alguns repositórios é de aproximadamente 20%. Já os pares de registros negativos estão sempre em maior quantidade.

Para esses 9 (nove) registros da Figura 4.3, por exemplo, faz-se necessário realizar 36 comparações. Dessa quantidade de comparações, apenas uma comparação leva a um exemplo positivo. Isso significa que aproximadamente 99,7% das comparações realizadas, ocorrem com registros negativos. Diante disso, pode se concluir que grande parte dos recursos computacionais gastos durante o treinamento de GP para deduplicação, se não houver nenhum tipo de pré trabalho anterior, são desnecessários.

### 4.2.4 Considerações sobre a Etapa de Treinamento

Em um *dataset*[1] contendo com 200 registros, por exemplo, será necessário realizar 19.900 comparações para cada *(g)* geração. Em termos de custo computacional para realizar o treinamento de um *dataset* com 200 registros, o tempo para comparação entre os registros é muito grande.

Vale ressaltar que alguns *datasets* possuem características diferentes, e isso influencia diretamente no tempo de comparação entre os registros. No caso da abordagem de [Carvalho et al., 2008a], eles utilizaram três *datasets*, sendo duas com dados reais e uma com dados sintéticos. Cada um desses *datasets* possui atributos específicos, e isso influencia diretamente no tempo para as comparações.

Para exemplificar, serão utilizados esses *datasets*, que são: *Restaurants*, *Cora* e *Synthetic* (descritas no Capítulo 5). O *dataset Restaurants*, possui 216 registros em

[1] é um conjunto de dados estruturados ou semiestruturados. Nesta pesquisa, opta-se por adotar o termo em inglês porque facilita a compreensão do objeto e, na literatura brasileira de Computação e Informática, seu uso é recorrente.

cada arquivo, o *dataset Cora* 200 registros em cada arquivo, e o *dataset Synthetic* 500 registros em cada arquivo.

Vale ressaltar que a principal característica do *dataset Cora* é a grande quantidade de *strings*[2] em seus atributos. O *dataset Synthetic* também possui muitas *strings* nos campos, porém em menor quantidade comparado ao *dataset Cora*. Já o *dataset Restaurants* possui poucas *strings* para descrição dos seus atributos. Isso influencia no custo computacional para avaliação dos *datasets*.

Isso é bem evidente nos resultados apresentados por [Carvalho et al., 2008a], onde a etapa de treinamento (que realiza comparação entre os registros) no *dataset Cora* custou aproximadamente 2.383 minutos, no *dataset Restaurants* foi 302 minutos e no *dataset Synthetic* esse custo foi de aproximadamente 1.885 minutos. Apesar do *dataset Cora* possuir menor quantidade de registros, o tempo de execução dela foi bem maior que os outros devido a quantidade de *strings* nos registros a serem comparados.

Outro exemplo que pode se destacar é o *dataset Synthetic*. Ele possui 04 arquivos com 500 registros cada. Um desses arquivos, tem apenas 49 registros originais e cada original possui de 1 a 4 réplicas. O total de comparações possíveis entre eles é 171. Porém, se a comparação for realizada entre todos os registros existentes no *dataset* (500), a quantidade de comparações será de 124.750. Isso significa uma diferença de 124.579 comparações que seriam realizadas entre registros negativos.

Visando minimizar esse problema, é proposta uma abordagem que reduz a quantidade de comparações desnecessárias, minimizando consideravelmente o tempo de treinamento, sem comprometer sua eficácia. Além disso, pode ser aplicado na maioria dos *dataset* onde se utilize a abordagem baseada em PG para deduplicação de registros. Essa técnica e os resultados obtidos serão descritos nos próximos Capítulos.

[2]é uma sequência de caracteres, geralmente utilizada para representar palavras, frases ou textos, definida pelo código ASCII.

# Capítulo 5

# Técnica de Agrupamento para a Etapa de Treino em PG

Neste capítulo, é descrita uma abordagem que utiliza uma técnica de agrupamento com o objetivo de agrupar registros similares próximos uns dos outros. Na seção 5.1 é detalhada a fundamentação para a abordagem proposta. Na seção 5.2 é descrita a abordagem para técnica de agrupamento.

## 5.1 Fundamentação para a Abordagem Proposta

A ideia principal desta proposta parte da abordagem de classificação que obedece as propriedades de transitividade das réplicas (descrita no Capítulo 4), de forma que, se um registro $A$ for réplica de um registro $B$ e $B$ for réplica de um registro $C$, então $A$ será réplica de $C$, e assim sucessivamente.

Assim, partindo dessa fundamentação, considerando essa transitividade, pode se concluir que existe um nível de similaridade entre os registros que são réplicas, disponíveis em um certo conjunto de dados.

Além disso, é utilizada a abordagem de [Christen, 2012], onde ressalta que para obtenção de bons resultados a qualidade dos dados de atributos em um registro é essencial, principalmente se não possuir valores vazios, ou não informados. O conceito foi aplicado no contexto de que pode se obter melhores resultados usando um registro contendo as mesmas características dos registros existentes na base de dados, porém com todos os campos preenchidos, para comparar com todos os registros existentes em um determinado conjunto de dados.

Assim, com base nesses conceitos, assume-se que as réplicas terão distancias equivalentes em relação ao registro referência, a partir da similaridade entre cada registro

do *dataset* e o registro referência.

## 5.2 Abordagem para Técnica de Agrupamento

A abordagem proposta por este trabalho visa diminuir o tempo de treinamento de PG, aplica os conceitos da técnica agrupamento baseada em particionamento (descrito no Capitulo 3). Os algorítimos de particionamento dividem os objetos entre os $k$ *clusters* de acordo com a medida de similaridade adotada, de modo que cada objeto fique no cluster que forneça o menor valor de distância entre o objeto e o centro do mesmo.

Para [Jain et al., 1999], o processo que esse algorítimo realiza, funciona da seguinte maneira: o primeiro passo é selecionar um registro (uma semente) como o centro inicial do grupo, e todos os objetos dentro de uma pré-determinada distância são incluídos no grupo resultante. Então, os objetos podem ser realocados se eles estiverem mais próximos de outro agrupamento do que daquele que originalmente lhes foi atribuído.

Escolhida a semente inicial (ou sementes), é calculada a distância de cada elemento em relação a semente, agrupando o objeto ao grupo que possuir a menor distância (mais similar).

Com base nisso, após vários testes (que serão descritos neste capítulo), essa técnica é utilizada da seguinte forma:

**Passo 1:** Escolhe um registro $r$ como o centro inicial do grupo.

**Passo 2:** Calcula a distância do registro $r$ para cada um dos outros $n$ registros utilizando uma função de similaridade.

**Passo 3:** Cria uma lista ordenada com base nas distâncias retornadas pela função de similaridade.

**Passo 4:** Utiliza a abordagem de Janela Deslizante (Descrita no Capítulo 6)) para comparar os registros que estiverem dentro da janela, obedecendo a lista ordenada no passo anterior.

Após esses passos, são realizadas as etapas de treinamento e testes para encontrar o melhor indivíduo (melhor F1) que será utilizado para a deduplicação do repositório, conforme a abordagem de [Carvalho et al., 2008a].

### 5.2.1 Abordagem Experimental

Para fundamentar cada passo da técnica de agrupamento proposta, a seguir descrevemos os principais conceitos necessários para o entendimento dela, e ao final apresentamos os resultados dos testes realizados.

#### 5.2.1.1 Bases de Dados Experimental

Em nossos experimentos foram utilizados três *datasets*. Dois deles possuem dados reais comumente empregados para avaliar abordagens relacionadas a deduplicação de registros ([Tejada et al., 2002]; [Bilenko, 2003]). Esses *datasets* são: *Cora* e *Restaurants*. Além disso, foi utilizado um *dataset* sintético (*Synthetic*), criado e usado por [Carvalho et al., 2008a].

O *dataset* Cora *(Cora Bibliographic Dataset)*, possui um conjunto de dados bibliográficos reais com 1295 citações diferentes de 122 *papers* da área de ciências da computação coletados da *web*. Essas citações foram divididas em múltiplos atributos *(authornames, year, title, venue, pagesandotherinfo)* por um sistema de extração de informações.

O *dataset* Restaurants, contém 864 entradas de nomes de restaurantes e algumas informações adicionais. Esse *dataset* possui 112 duplicatas, que foram obtidos integrando os registros de Fodor e Zagat's *guidebooks*. Foram utilizados os seguintes atributos: *(name, address, city, specialty)*.

O *dataset Synthetic*, foi criado por [Carvalho et al., 2008a] utilizando o *Synthetic Dataset Generator* (SDG) do *Febrl* método proposto por [Christen, 2007]. Isso porque dados reais não são facilmente disponíveis para a experimentação, devido a restrições de privacidade e confidencialidade. Assim, a fim de realizar um conjunto maior de experimentos para avaliar a sua abordagem, esse *dataset* foi criado contendo 2000 registros no total. Ele possui seguintes atributos: *(forename, surname, street number, address1, address2, suburb, postcode, state, date of birth, age, phone number, social security number)*.

Em seus experimentos, [Carvalho et al., 2008a] dividiu cada *dataset* em 4 (quatro) *datasets* menores. Em nossos experimentos, a quantidade de total de registros e a distribuição em *datasets* menores que utilizamos, estão dispostas de acordo com Tabela abaixo:

**Tabela 5.1.** Características dos *datasets* utilizados em nossos experimentos.

| **Nome do *dataset*** | **Qtd de Registros** | ***dataset* 1** | ***dataset* 2** | ***dataset* 3** | ***dataset* 4** |
|---|---|---|---|---|---|
| Cora | 800 | 200 | 200 | 200 | 200 |
| Synthetic | 2000 | 500 | 500 | 500 | 500 |
| Restaurants | 864 | 216 | 216 | 216 | 216 |

#### 5.2.1.2 Abordagem para Agrupamento de Registros Similares

A partir da abordagem de classificação que obedece as propriedades de transitividade das réplicas (descrita na seção 5.1), foram utilizadas três estratégias para escolha do *registro referência*. Isto é, o registro que será comparado com todos os outros do *dataset*. Esse registro será usado com objetivo de definir a forma mais efetiva para realizar o agrupamento de registros similares. Assim, foi testado o registro referência da seguinte forma:

***1) Sintético:*** É um registro criado a partir da junção entre atributos de registros diferentes existentes no *dataset* a ser avaliado (de acordo com a quantidade de atributos de cada registro existente em cada *dataset*).

***2) Aleatório:*** É escolhido um registro aleatoriamente dentre os registros existentes no *dataset* (sem ser previamente analisado).

**3) *String ruído:*** Trata-se de um registro com as mesmas características do domínio dos tipos de dados existentes no *dataset*, contendo todos os campos preenchidos. (descrito na subseção 5.2.1.5).

A fundamentação teórica para a criação dos registros referência, de como correu a concatenação de *strings* e os testes realizados, serão apresentados nas próximas subseções.

#### 5.2.1.3 Registro Sintético

Nesta seção será apresentada a fundamentação para criação do registro referência Sintético. Esse registro deve possuir todos os campos preenchidos por atributos existentes nos registros do *dataset*.

Para isso foi realizada uma escolha de vários registros de acordo com a quantidade de atributos de cada registro. Em seguida, foi criado um novo registro contendo a junção de vários atributos dos registros do *dataset*.

Podemos exemplificar utilizando o *dataset Restaurants*. Nesse *dataset* cada registro possui 6 (seis) atributos. Assim, o registro sintético foi criado a partir da utilização de um atributo de cada um dos seis registros.

#### 5.2.1.4 Registro Aleatório

É um registro escolhido de forma aleatória dentre todos os registros existentes no *dataset* a ser avaliado. Essa escolha ocorre com base na quantidade de registros existentes no *dataset*.

Para exemplificar, utilizaremos o *dataset Restaurants*. Esse *dataset* possui 216 (duzentos e dezesseis) registros. Neste caso, o *dataset* é carregado em uma lista e um registro é aleatoriamente escolhido dentre os 216 (duzentos e dezesseis).

#### 5.2.1.5 Fundamentação Teórica para a *String Ruído*

Nesta seção será apresentada a fundamentação para criação do registro referência utilizando a *string ruído*. De modo geral, esse registro tem como objetivo representar o domínio dos tipos de atributo de uma determinada entidade. Para isso leva se em consideração as seguintes definições:
**1)** *Entidade:* é uma "coisa" ou um "objeto" no mundo real que pode ser identificada de forma unívoca em relação a todos os outros objetos [Silberschatz et al., 2012]. Por exemplo, cada servidor de uma instituição pública de ensino é uma entidade. Cada unidade de ensino (campus) desse órgão também.
**2)** *Conjunto de Entidades:* é uma coleção de entidades que têm características semelhantes, isto é, de entes de uma mesma categoria [Elmasri, 2013]. Para exemplificar, o conjunto de entidades Servidores representa a coleção de todos os servidores que trabalham naquela instituição. E o conjunto de entidades Campi refere-se ao conjunto de todas as unidades de ensino daquele órgão.
**3)** *Atributo:* são propriedades descritivas de cada membro de um conjunto de entidades [Silberschatz et al., 2012]. Em outras palavras, atributos são os dados que se deseja guardar sobre cada entidade. Desta forma, o nome, o prontuário e a data de nomeação são possíveis atributos para cada entidade do conjunto de entidades Servidores. Endereço e sigla comporiam os atributos de cada ente do conjunto de entidades Campi.
**4)** *Domínio:* é o tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna, ou seja, um conjunto de valores válidos para um determinado atributo [Elmasri, 2013]. Um atributo "tira seus valores de um conjunto de valores correspondente (isto é, domínio, em outras palavras)" [Elmasri, 2013]. Por exemplo, em determinada instituição pública o servidor pode, conforme o cargo ocupado, cumprir uma jornada de trabalho de 20, 30 ou 40 horas semanais. Considerando que Carga-Horária seja um dos atributos de Servidores, o conjunto formado pelos valores 20, 30 e 40 compõem o domínio dessa propriedade, uma vez que Carga-Horária tem, obrigatoriamente, que assumir um desses três valores.

Com base nessas definições, parte-se da abordagem que os registros existentes nos repositórios são compostos por vários atributos, e cada atributo é composto por um domínio diferente. Assim, a criação da *string ruído* para um certo tipo de domínio não fica restrita aos valores do domínio, mas ao tipo de dados utilizado na representação

daquele domínio.

Desse modo, o registro referência *string ruído* será criado manualmente a partir da observação dos tipos de dados existentes nos domínios dos registros do *dataset* a ser avaliado. Assim, cada atributo desse registro será representado apenas pelo tipo de dados e não pelo valor do domínio como ocorre nos registros aleatório e sintético.

Para exemplificar, consideremos que o tipo de domínio para nome de pessoas normalmente codificado na forma de *string*. Para ela, será então gerada uma *string* com caracteres aleatória, não necessariamente pertencendo ao conjunto de valores daquele domínio. Digamos que uma instância do atributo nome do servidor seja "Jeremias Carneiro", um exemplo de *string ruído* aleatória para esse domínio seria "AAAAA BBBBB", e assim por diante. Do mesmo modo, se no caso o tipo de domínio for numérico (Idade, Índices), será gerado um número aleatório dentro de uma faixa de dados, porém não receberá os mesmos valores contidos nesses atributos.

Na Figura 5.1 é mostrado um exemplo de utilização da *string ruído*.

**Tabela Cliente**

Atributos do registro

| | Nome do Cliente | Contato | Telefone fixo | Estado |
|---|---|---|---|---|
| | Ana Maria Well | Ana Well | 09233445577 | AM |

String ruído

| | Nome do Cliente | Contato | Telefone fixo | Estado |
|---|---|---|---|---|
| | Cccc bBb | gff Fnn | 55555 | JJ |

**Figura 5.1.** Exemplo de *String ruído*.

#### 5.2.1.6 Comparação de registros

Para realizar a comparação entre o registro referência e cada registro dos *datasets*, tornou-se necessário utilizar algoritmos de cálculo de similaridade para comparação entre os registros. Neste trabalho utilizaremos as mesmas funções de similaridade usadas por [Carvalho et al., 2006] em sua abordagem.

De acordo com [Carvalho et al., 2006] as funções de similaridade ou algoritmos de similaridade $f(a_1, a_2) \mapsto s$ calculam quanto uma entrada $a_1$ é similar a outra $a_2$. Tais algoritmos determinam um escore $s$ para cada par de valores de dados (*strings*). Escores altos significam similaridade alta.

Técnicas para similaridade de campos complexos[1] utilizam um conjunto de téc-

[1]Conjuntos (tuplas) ou coleções de valores.

nicas de valores atômicos[2], combinando seus escores resultantes para que se possa ter apenas um valor de similaridade para o campo complexo. Por exemplo, precisa-se saber a similaridade entre duas tuplas, $T_1$ e $T_2$, cada uma dessas tuplas possui dados de uma pessoa, como nome, endereço e telefone [Tejada et al., 2001].

A maior parte das técnicas de similaridade de registros conhecidas (campos complexos) utilizam uma função de similaridade para os atributos separadamente, como por exemplo, nome, endereço e telefone, e então fazem a combinação desses valores para obter um valor de similaridade entre as tuplas. Em contrapartida outras técnicas concatenam[3] todos os campos em apenas um (formando apenas uma *string*) e verificam a similaridade da *string* inteira [Tejada et al., 2001].

Porém, [Tejada et al., 2001] dizem que concatenar campos e verificar a similaridade como uma *string* apenas, gera imprecisão. Estudos sobre esse assunto são encontrados em [Tejada et al., 2001], [Elmagarmid et al., 2007] e [Guha et al., 2004].

Com base nisso, em nossa abordagem, a comparação entre *strings* foi realizada sem concatenação, ou seja, cada atributo foi comparado separadamente e então foi feita uma combinação das similaridades retornadas para obter o valor de similaridade entre o registro referência e cada registro do *dataset*.

A combinação das similaridades que utilizamos foi a mesma usada por [Carvalho et al., 2006], onde foram somadas todas as similaridades e foi tirada a média pela quantidade de atributos dos registros.

#### 5.2.1.7 Aproximando os Registros Originais de suas Réplicas

Nesta seção será explicada a estratégia que se utilizou para agrupar um registros próximos de suas réplicas. Porém, é importante destacar que em todos os *datasets* um registro original pode ter vários registros réplicas ou nenhum, todos previamente identificados [Carvalho et al., 2008a].

O primeiro passo para o agrupamento é que para cada *dataset* foi escolhido uma das estratégias para o registro referência e uma função de similaridade. Para cada *dataset* será aplicada uma função $f$ em todos os pares de registro possíveis formados entre o registro referência e todos os outros registros do *dataset*. Terminada essa etapa, os registros foram ordenados por similaridade em ordem crescente.

Para cada registro do *dataset* que possui réplica, foi calculada a distância desse registro para sua(s) réplica(s). A partir desse cálculo, geramos as seguintes métricas:

[2]São dependentes do domínio da aplicação, ou seja, consideram as características dos dados da aplicação.

[3]é a junção de duas ou mais palavras, ou seja, a operação de unir o conteúdo de duas *strings*.

*Dmax*- Distância máxima encontrada entre um registro e qualquer uma de suas réplicas;
*Dmed*- Distância média entre um registro e todas as suas réplicas.

Feito isso é calculada a média das distâncias máximas e a média das distâncias médias para definir a função de similaridade que melhor agrupe um registros e suas réplicas, obtendo a menor *Dmax*.

Na Figura 5.2 é mostrado um exemplo do agrupamento de registros após a aplicação de uma função de similaridade.

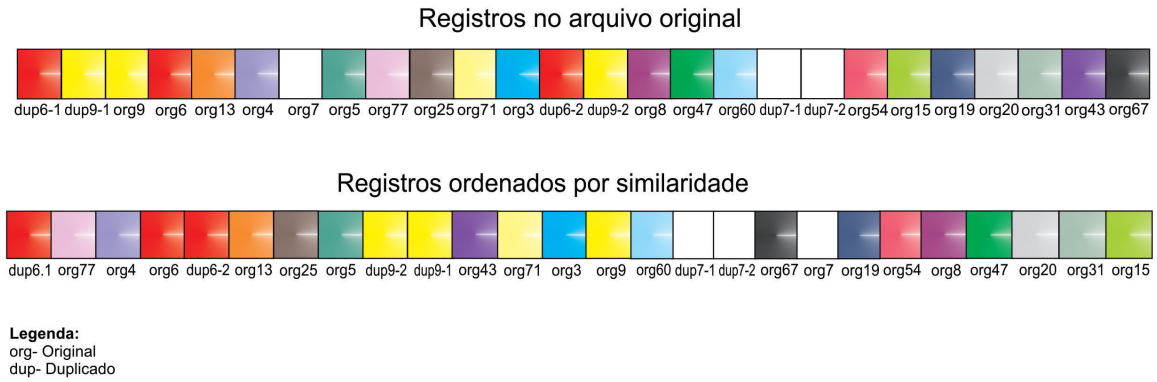


**Figura 5.2.** Divisão de um arquivo em *clusters*.

Reforçando que o objetivo da nossa métrica é a partir dessa comparação colocar os registros similares próximos uns dos outros.

#### 5.2.1.8 Experimento para Definição da Função de Similaridade

Visando encontrar uma função de similaridade que melhor agrupasse os registros similares, foram realizado vários testes. Cada função de similaridade foi aplicada no *dataset*, e na saída buscou-se a função que obtivesse o menor valor em relação a maior distância entre um registro e sua(s) réplica(s) e a distância média entre um registro e sua(s) réplica(s).

Esses testes, foram aplicados nas três opções de classificação apresentadas na subseção 5.2.1.2 anterior. A estratégia experimental para treino e as funções de similaridades usadas para estes experimentos foram as mesmas utilizadas por [Carvalho et al., 2008a].

As funções de similaridade utilizadas na abordagem proposta por [Carvalho et al., 2008a] foram: *Levenshtein distance* (Editdist), cosseno de similaridade (Softtfidf), Jaro, Sortwinkler, Winkler, Bigrama, Bagdist, combinação de sequência (Seqmatch), e Compression.

Na Tabela 5.2 são apresentados os resultados de *Dmax* e *Dmed* para cada função de similaridade aplicada no *dataset Restaurants*.

**Tabela 5.2.** Aplicação das funções de similaridade no *dataset Restaurants*.

| **Resultados no *dataset Restaurants*** | | | | | | |
|---|---|---|---|---|---|---|
| **Função** | Distância *String* Ruido | | Distância Registro Sintético | | Distância Registro Aleatório | |
| | Dmax | Dmed | Dmax | Dmed | Dmax | Dmed |
| Editdist | **51** | **27** | 62 | 29 | 52 | 31 |
| Bagdist | 61 | 35 | 64 | 36 | 61 | 34 |
| Jaro | 62 | 37 | 78 | 31 | 64 | 38 |
| Sortwinkler | 69 | 27 | 99 | 58 | 85 | 41 |
| Bigram | 72 | 33 | 155 | 46 | 75 | 36 |
| Softtfidf | 84 | 29 | 57 | 29 | 52 | 33 |
| Seqmatch | 89 | 45 | 201 | 62 | 52 | 28 |
| Winkler | 91 | 44 | 139 | 55 | 97 | 41 |
| Compression | 129 | 48 | 105 | 42 | 112 | 40 |

Para o *dataset Restaurants* com a utilização do registro referência com *string ruído*, apresentou melhor resultado com a menor distância entre as distância máxima entre um registro original e suas réplicas.

De maneira análoga os mesmos testes foram realizados no *dataset* Cora, e os resultados são apresentados na Tabela 5.3.

**Tabela 5.3.** Aplicação das Funções de similaridade no *dataset Cora*.

| **Resultados no *dataset Cora*** | | | | | | |
|---|---|---|---|---|---|---|
| **Função** | Distância *String* Ruido | | Distância Registro Sintético | | Distância Registro Aleatório | |
| | Dmax | Dmed | Dmax | Dmed | Dmax | Dmed |
| Bigram | **97** | **23** | 122 | 55 | 123 | 63 |
| Bagdist | 104 | 41 | 118 | 49 | 103 | 41 |
| Compression | 104 | 44 | 176 | 51 | 140 | 48 |
| Seqmatch | 108 | 42 | 151 | 55 | 107 | 42 |
| Softtfidf | 113 | 56 | 149 | 33 | 103 | 41 |
| Sortwinkler | 131 | 60 | 176 | 51 | 126 | 50 |
| Jaro | 149 | 60 | 138 | 47 | 138 | 47 |
| Editdist | 149 | 63 | 138 | 47 | 128 | 61 |
| Winkler | 152 | 56 | 113 | 60 | 127 | 41 |

No *dataset Cora*, a utilização do registro referência com *string ruído*, apresentou melhor resultado com a menor distância entre as distância máxima entre um registro original e suas réplicas.

O terceiro *dataset* a ser utilizado para os testes foi a *dataset* Synthetic, e os resultados são apresentados na Tabela 5.4.

**Tabela 5.4.** Aplicação das Funções de similaridade no *dataset Synthetic*.

| **Resultados no *dataset Synthetic*** | | | | | | |
|---|---|---|---|---|---|---|
| **Função** | Distância *String* Ruido | | Distância Registro Sintético | | Distância Registro Aleatório | |
| | Dmax | Dmed | Dmax | Dmed | Dmax | Dmed |
| Seqmatch | **162** | **28** | 322 | 84 | 322 | 84 |
| Jaro | 166 | 29 | 312 | 119 | 312 | 119 |
| Winkler | 167 | 28 | 291 | 122 | 291 | 122 |
| Sortwinkler | 177 | 30 | 312 | 119 | 318 | 124 |
| Editdist | 209 | 41 | 399 | 287 | 287 | 65 |
| Bagdist | 236 | 41 | 399 | 282 | 282 | 67 |
| Compression | 282 | 75 | 357 | 44 | 357 | 44 |
| Bigram | 385 | 30 | 311 | 93 | 311 | 93 |
| Softtfidf | 445 | 150 | 277 | 54 | 277 | 171 |

Semelhantemente aos *datasets* testados anteriormente, a utilização do registro referência com *string ruído*, apresentou melhor resultado com a menor distância entre as distância máxima entre um registro original e suas réplicas, no *dataset Synthetic*.

#### 5.2.1.9 Avaliação dos Resultados Preliminares

Os experimentos no *dataset Restaurants* utilizando *string ruído* apresentaram resultados equivalentes ao registro aleatório na maioria das funções de similaridade. Porém, nos *datasets Cora* e *Synthetic*, os resultados obtidos pela *string ruído* apresentaram valores de *Dmax* menores. Esses resultados estão relacionados às principais características dos *datasets* (descritas no capítulo 4) onde a quantidade de *strings* em seus atributos influencia tanto no custo de processamento quanto na comparação entre os registros.

Baseado nos experimentos realizados, conclui-se que usar a *string ruído* é a melhor abordagem não importa o *dataset*, nem a função de similaridade que seja utilizada. Além disso, não existe uma função de similaridade que seja melhor em todos os *datasets*, ou seja, para cada *dataset* a função de similaridade pode ser diferente.

Vale ressaltar que conforme resultados apresentados, a partir da *string ruído* obtêm-se sempre as menores *Dmax*. Assim, a função que retorna a menor *Dmax* será aplicada para a avaliação do *dataset*.

Isso vai ao encontro das definições de [Christen, 2012] onde ressalta que a qualidade dos dados de atributos em um registro é essencial para obtenção de bons resulta-

dos. Quando o atributo possui muitos valores vazios, ou não informados, pode ocorrer que muitos registros comparados ditos similares, na prática não o são. Se há muitos erros pode ocorrer que registros similares sejam distribuídos incorretamente.

E apesar da aplicação dessas definições feitas por [Christen, 2012] ser utilizada em outro problema, os resultados pela abordagem demonstraram que os dados se comportam de forma muito parecida. Com base nesses resultados, pode-se inferir que a abordagem de escolha do registro referência utilizando *string ruído* é a mais adequada para a maioria dos domínios de dados.

Assim, optou-se por utilizar a *string ruído*, não apenas pelos melhores resultados, mas por observar que em alguns *datasets* é comum existir campos com valores nulos ou incorretos. Com base nessas definições e nos resultados obtidos, fundamentou-se a escolha da *string ruído* e funções de similaridade utilizadas nos passos 1 e 2, seção 5.2.

# Capítulo 6

# Estratégia para Comparação de Registros

Neste capítulo é detalhada a aplicação de uma abordagem que utiliza uma estratégia de janela deslizante para comparação entre os registros já ordenados por uma função de similaridade. São apresentados também os resultados de um estudo experimental que mostra a aplicação dessa abordagem na etapa de treinamento em PG. Além disso, será feita uma avaliação dessa abordagem visando identificar quais as situações que facilitam ou dificultam a utilização da estratégia proposta.

## 6.1 Estratégia de Janela Deslizante

Após o resultado retornado pela similaridade entre o registro referência e os outros registros, realizou-se o agrupamento de todos os $n$ registros em uma lista ordenados pelo resultado da similaridade. O próximo passo é a comparação de todos os registros entre si durante a etapa de treino da PG.

Para realizar essa comparação, foi adotada uma estratégia de janela deslizante que possibilitará a comparação entre todos os registros que estiverem em um intervalo que será definido pelo tamanho dessa janela.

A estratégia de janela deslizante foi aplicada com base na abordagem de [Ziv et al., 1977], onde os autores propuseram um modelo de "Janela Deslizante" de tamanho fixo ($w > 1$), que move-se sequencialmente sobre os registros ordenados a um deslocamento ($v$). Todos os registros que estiverem dentro da mesma janela serão comparados. Ao final, cada registro gerará ($2w$ - 1) pares para comparação, resultando um total de $O(nw)$ pares em um *dataset* com $n$ registros no total.

A principal vantagem desta técnica é que a quantidade de pares comparados pode ser controlada. Todos os registros que estiverem no intervalo definido pelo tamanho da janela, são comparados e incluídos na lista de réplicas ou não-réplicas. A velocidade $v$ tem a função de definir a quantidade de linhas de registros que a janela deslizará a cada interação.

Essa estratégia será aplicada após a comparação de registros utilizando o registro referência com todos registros do *dataset* (Capítulo 5). O cálculo da distância entre os registros é feito por uma função de similaridade e a partir desses resultados espera-se que os registros similares estejam próximos.

Vale ressaltar que é de suma importância a observação durante o processo de configuração do tamanho da janela e da velocidade de deslizamento. Isso torna-se essencial para um resultado satisfatório na etapa de treinamento da PG. Porém, se essa configuração não for realizada corretamente, os resultados poderão ser ruins e o tempo de processamento aumentará consideravelmente.

Utilizando essa estratégia, na primeira execução com determinado tamanho de janela e deslocamento, todos os registros entre sí serão comparados, ou seja, $(\frac{n\times(n-1)}{2})$ comparações de registros referentes ao tamanho da janela. A partir da segunda execução da janela, os registros já comparados não serão comparados novamente entre si, porém os registros que estavam fora da janela serão comparados com os anteriores de acordo com o tamanho da janela.

Isso significa que a quantidade de comparações entre os registros diminuirá consideravelmente, tendo impacto significativo no tempo final de execução. Outro fator importante a se destacar é a diminuição de comparações desnecessárias entre registros não similares.

A aplicação da estratégia que utiliza janela deslizante em nossa abordagem, foi realizada conforme exemplo apresentado na Figura 6.1. Nesse exemplo, foi definida uma janela de tamanho 10 e velocidade de deslizamento com valor 3.

O item **1** da Figura 6.1 mostra os registros que foram ordenados baseado na sua similaridade em relação ao registro referência. Espera-se que os registros similares estejam próximos uns dos outros.

Os itens **2, 3, 4 e 5**, mostram a janela deslizando e todos os registros que estão no intervalo definido pelo tamanho janela serão comparados entre si. Assim, é grande a possibilidade de encontrar registros réplicas nesses intervalos.

Nas próximas seções apresentados os resultados obtidos por meio dessa abordagem em *datasets* reais e sintéticos, utilizando a aplicação prática da estratégia de janela deslizante.

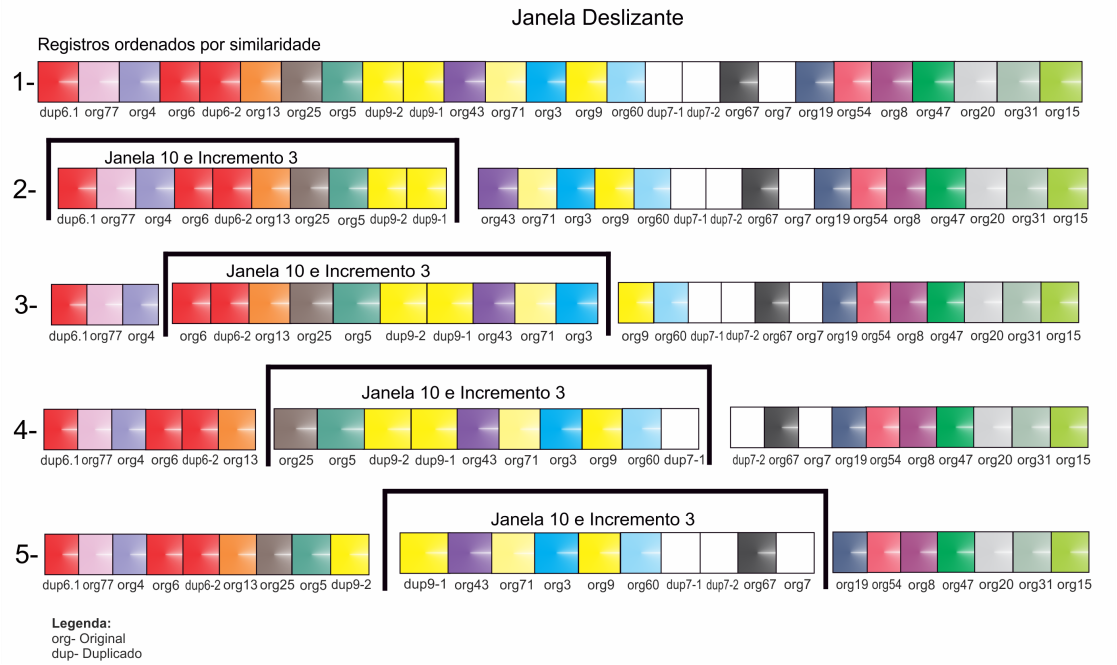


**Figura 6.1.** Exemplo da estratégia de janela deslizante.

## 6.2 Configuração Experimental

Para realizar os experimentos, foram utilizadas configurações padrões da abordagem de [Carvalho et al., 2008a] para todas os *datasets*. Os *datasets* são os mesmos descritos no Capítulo 5 e também utilizados nos experimentos de [Carvalho et al., 2008a]. As configurações de PG serão descritas na Tabela 6.2.

**Tabela 6.1.** Principais parâmetros de PG utilizados.

| Parâmetros | Valor |
|---|---|
| Número de Execuções | 20 |
| Tamanho da População | 100 |
| Número Máximo de Gerações | 20 |
| Método de Geração da População Inicial | (1)FullDepth |
| Método de Pareamento dos Indivíduos | (1)Random |
| Método de Seleção | (2)Roullete Wheel |
| Profundidade Máxima da Árvore Random | 4 |

Baseado nos resultados obtidos pela métrica *Dmed* (Capítulo 5), para os experimentos realizados com o registro referência *string ruído*, foi definido para nossa experimentação utilizar janelas inicialmente com aproximadamente metade do menor valor das *Dmed* obtidas em todos os *datasets*. Do mesmo modo, o tamanho do maior valor para janela, seria um pouco acima do valor da média de todas as *Dmed* dos *datasets*. Foram utilizados 05 tamanhos de janelas diferentes para os testes.

Para exemplificar, o menor valor de todas as *Dmed* dos *datasets* foi encontrado no *dataset Cora* com valor 23. Assim, a metade desse valor seria aproximadamente 12. Com isso, foi definido o valor da janela inicial para ser aplicado em todos os *datasets*. Já o valor do maior tamanho de janela foi de aproximadamente 71.

Além disso, foi decidido utilizar os valores dos deslocamentos (velocidade da janela) baseados na porcentagem da quantidade de registros em cada *dataset*. Essas porcentagens foram variadas em uma média de aproximadamente 3% a 10% em relação a quantidade de registros em cada *dataset*. Do mesmo modo que foram realizados os testes com as janelas, também utilizou-se 05 valores de deslocamento diferentes.

Utilizando como base o mesmo exemplo descrito acima, o valor do deslocamento inicial baseado na porcentagem da quantidade de registros (200) existente *dataset Cora* foi 7 e o valor final foi 23.

Esses testes tiveram como objetivo avaliar o impacto de qualidade e tempo gasto para cada *dataset*, variando tanto o tamanho da janela quanto o valor do deslocamento.

Na Tabela 6.2 são apresentados os tamanhos de janela e deslocamento que foram utilizadas para os testes em todos os *datasets*, a partir da abordagem explicada anteriormente.

**Tabela 6.2.** Configurações da Janela e Deslocamento

| |
|---|
| Janela 12 e Deslocamento 7 |
| Janela 23 e Deslocamento 7 e 11 |
| Janela 41 e Deslocamento 7, 11 e 15 |
| Janela 58 e Deslocamento 7, 11, 15 e 19 |
| Janela 71 e Deslocamento 7, 11, 15, 19 e 23 |

Esses valores assumidos pela janela e deslocamento, foram utilizados para demonstrar o comportamento da Precisão, Revocação, F1 e Acurácia; durante o treino e teste. Para os *datasets* que possuem valores de distância máxima entre os registros que são acima de 71, foi fixado um tamanho de janela e deslocamento um pouco acima do valor retornado, que será descrito no final de cada tabela, para observar a efetividade desta técnica.

Tendo em vista que o maior tamanho definido para Janela deslizante 71 e que o valor da média das *Dmax* no *dataset Synthetic* foi 162, decidiu-se fazer um teste utilizando uma janela de tamanho 175 e deslocamento 7. Do mesmo modo, o valor da média das *Dmax* do *dataset Cora* foi 97, decidiu-se fazer um teste utilizando uma janela de tamanho 105 e deslocamento 7. Os testes com esses tamanhos de Janela foram apenas para mostrar que pode-se ter bons resultados se a análise das distâncias obtidas pelas funções de similaridade forem observadas.

Os experimentos utilizando as configurações descritas na Tabela 6.2 foram realizados em três computadores *Desktop* contendo as mesmas configurações, conforme Tabela 6.3 abaixo:

**Tabela 6.3.** Configurações do computador.

| Sistema Operacional | Processador | Memória RAM (GB) |
|---|---|---|
| Linux Ubuntu 15.05 - 64Bits | $Core^{TM}$ i5 | 8 |

## 6.3 Métricas de Avaliação

Em nossos experimentos foram aplicadas as mesmas métricas de avaliação utilizadas na abordagem de [Carvalho et al., 2008a], que são: Precisão, Revocação e F1 (Descritas no Capítulo 4 seção 4.2.2).

A métrica Precisão é responsável por mensurar a proporção de réplicas identificadas corretamente dentre todas as identificações realizadas; a Revocação é utilizada para calcular a proporção de réplicas identificadas corretamente dentre todas as identificações que deveriam ter sido feitas; e a F1 é a métrica capaz de combinar precisão e revocação, ou seja, a função de *fitness*.

## 6.4 Avaliação Experimental da Técnica Proposta

Nas subseções seguintes serão descritos os resultados obtidos nos experimentos realizados em cada *dataset* separadamente. Esses resultados serão sempre comparados com os obtidos pela abordagem de [Carvalho et al., 2008a] que utilizamos como *baseline*, e estão descritos nas tabelas como *PG configuração padrão*.

Para cada um dos três *datasets* utilizados, criamos duas tabelas. A primeira tabela apresenta os resultados relacionados a Precisão, Revocação e F1, tanto na etapa de treinamento quanto na etapa de testes.

A segunda tabela apresenta os resultados do Tempo Gasto no Treino (em segundos), a Porcentagem do Tempo em relação ao *baseline*, o Total de Comparações entre registros em cada configuração e a Quantidade de Réplicas Identificadas corretamente.

A partir do resultado da aplicação do registro referência *string ruido* com os registros existentes em cada *dataset*, foi encontrada a função que melhor agrupou os registros similares. Após a aplicação dessa função, foi utilizado a abordagem proposta para comparação dos registros. Os resultados serão descritos nas tabelas a seguir.

### 6.4.1 Experimentos com o *dataset Restaurants*

Na Tabela 6.4 abaixo, são descritos os resultados obtidos pela técnica proposta tanto na etapa de treino quanto na etapa de teste. A função que melhor agrupou registros similares no *dataset Retaurants* foi a função Distância Levenshtein ou distância de edição (representado na abordagem de [Carvalho et al., 2008a] por "EditdIst").

**Tabela 6.4.** Resultados dos experimentos no *dataset Restaurants*

| **Resultados: *dataset Restaurants*** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Configurações Gerais | | Treino | | | Teste | | |
| Parâmetros | | Precisão | Revocação | F1 ($\sigma$) | Precisão | Revocação | F1 ($\sigma$) |
| PG configuração padrão | | 1.000 | 1.000 | 1.000 ±(0.000) | 1.000 | 0.963 | 0.981±(0.019) |
| Tam. janela | Deslocamento | | | | | | |
| 12 | 7 | 1.000 | 0.570 | 0.726 ±(0.218) | 1.000 | 0.535 | 0.694±(0.236) |
| 23 | 7 | 1.000 | 0.801 | 0.889 ±(0.100) | 1.000 | 0.800 | 0.826±(0.109) |
| | 11 | 1.000 | 0.596 | 0.747 ±(0.204) | 1.000 | 0.580 | 0.739±(0.212) |
| 41 | 7 | 1.000 | 0.779 | 0.876 ±(0.111) | 1.000 | 0.713 | 0.858±(0.144) |
| | 11 | 1.000 | 0.750 | 0.857 ±(0.125) | 1.000 | 0.750 | 0.778±(0.137) |
| | 15 | 1.000 | 0.775 | 0.873 ±(0.113) | 1.000 | 0.750 | 0.856±(0.125) |
| 58 | 7 | 1.000 | 0.953 | 0.976 ±(0.024) | 1.000 | 0.930 | 0.949±(0.036) |
| | 11 | 1.000 | 0.889 | 0.941 ±(0.056) | 1.000 | 0.860 | 0.887±(0.076) |
| | 15 | 1.000 | 0.810 | 0.895 ±(0.095) | 1.000 | 0.792 | 0.871±(0.105) |
| | 19 | 1.000 | 0.792 | 0.884 ±(0.104) | 1.000 | 0.698 | 0.813±(0.152) |
| 71 | 7 | 1.000 | 0.997 | 0.988 ±(0.012) | 1.000 | 0.949 | 0.975±(0.026) |
| | 11 | 1.000 | 0.937 | 0.967 ±(0.032) | 1.000 | 0.900 | 0.947±(0.050) |
| | 15 | 1.000 | 0.868 | 0.929 ±(0.066) | 1.000 | 0.829 | 0.898±(0.086) |
| | 19 | 1.000 | 0.804 | 0.872 ±(0.034) | 1.000 | 0.791 | 0.883±(0.105) |
| | 23 | 1.000 | 0.829 | 0.907 ±(0.086) | 1.000 | 0.802 | 0.889±(0.099) |

Os melhores resultados obtidos foram a partir das janelas de tamanho 71 e 58 com valor de deslocamento 7, tanto na revocação quanto no treino.

Um comentário importante a ser feito é que a precisão é sempre 1.0, o que significa que o treinamento não comete erro de identificação dos pares replicados. Ter diminuído a quantidade de instrução do treino não diminuiu a precisão. Apesar de ter reduzido a quantidade exemplos, o método aprendeu com os exemplos existentes na janela dada.

Vale ressaltar também que nesse *dataset* a precisão não foi influenciado pelo tamanho da janela e deslocamento. Já a revocação teve influência porque quanto maior a janela mais exemplos havia para o treinamento da PG e conseguia aprender mais. A tendencia observada é que quanto mais aumenta o tamanho da janela normalmente se tem um resultado melhor.

Na Tabela 6.5 são apresentados os resultados referentes ao tempo gasto na etapa de treinamento, a porcentagem em comparação ao *PG configuração padrão* e a quantidade de réplicas encontradas.

**Tabela 6.5.** Resultados dos experimentos no *dataset Restaurants*

| **Resultados: *dataset Restaurants*** | | | | | |
|---|---|---|---|---|---|
| Configurações (*Dataset* com 216 Registros) | | Tempo do Treino (Segundos) | | Total de Comparações | Réplicas Identificadas |
| (Total de réplicas no *dataset*: 10) | | Total | %Tempo | Qtd | Qtd |
| PG configuração padrão | | 4.466.850 | 100% | 23.220 | 10 |
| Tam. janela | Incremento | | | | |
| 12 | 7 | 148.994 | 3.3% | 1.695 | 7 |
| 23 | 7 | 341.260 | 7.6% | 3.860 | 8 |
| | 11 | 317.740 | 7.1% | 3.444 | 7 |
| 41 | 7 | 683.190 | 15.3% | 7.295 | 8 |
| | 11 | 805.695 | 18.0% | 6.625 | 8 |
| | 15 | 592.770 | 13.3% | 6.291 | 8 |
| 58 | 7 | 109.5330 | 24.5% | 10.020 | 9 |
| | 11 | 104.1845 | 23.3% | 9.708 | 9 |
| | 15 | 881.510 | 19.7% | 9.196 | 8 |
| | 19 | 823.640 | 18.4% | 8.988 | 8 |
| 71 | 7 | 1.214.322 | 27.2% | 11.929 | 10 |
| | 11 | 1.199.585 | 26.9% | 11.840 | 10 |
| | 15 | 1.177.405 | 26.4% | 11.046 | 9 |
| | 19 | 1.130.820 | 25.3% | 10.650 | 7 |
| | 23 | 953.950 | 21.4% | 10.675 | 9 |

Observando o resultado do tempo, a janela de tamanho 58 com deslocamento 7, realizou o processo com 24.5% do tempo comparado ao *PG configuração padrão* encontrando 9 das 10 réplicas. Já a janela com tamanho 71 com deslocamento 7 realizou com 27.2% do tempo e encontrou todas as réplicas existentes no *dataset*.

### 6.4.2 Experimentos com o *dataset Synthetic*

Neste *dataset* a função que melhor agrupou registros similares foi a função de combinação de sequência (Seqmatch [Carvalho et al., 2008a]).

Na Tabela 6.6 abaixo, são descritos os resultados obtidos pela técnica proposta tanto na etapa de treino quanto na etapa de teste.

Semelhantemente aos resultados do *dataset Restaurants*, os melhores resultados obtidos no *dataset Synthetic* foram a partir das janelas de tamanho 58 e 71 com valor de deslocamento 7, tanto na revocação quanto no treino.

É importante ressaltar que quanto mais aumenta o tamanho da janela os resultados tendem a melhorar, porém ao aumentar o valor do deslocamento os resultados são

**Tabela 6.6.** Resultados dos experimentos no *dataset Synthetic*

| **Resultados: *dataset Synthetic*** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Configurações Gerais | | Treino | | | Teste | | |
| Parâmetros | | Precisão | Revocação | F1 ($\sigma$) | Precisão | Revocação | F1 ($\sigma$) |
| PG configuração padrão | | 0.988 | 0.953 | 0.970±(0.017) | 0.967 | 0.928 | 0.947 ±(0.020) |
| Tam. janela | Deslocamento | | | | | | |
| 12 | 7 | 0.843 | 0.743 | 0.789±(0.050) | 0.835 | 0.684 | 0.752 ±(0.076) |
| 23 | 7 | 0.851 | 0.784 | 0.816±(0.034) | 0.843 | 0.763 | 0.801 ±(0.040) |
| | 11 | 0.872 | 0.755 | 0.809±(0.059) | 0.872 | 0.687 | 0.769 ±(0.093) |
| 41 | 7 | 0.996 | 0.794 | 0.884±(0.101) | 0.996 | 0.757 | 0.860 ±(0.120) |
| | 11 | 1.000 | 0.802 | 0.827±(0.099) | 1.000 | 0.716 | 0.834 ±(0.143) |
| | 15 | 0.997 | 0.791 | 0.882±(0.103) | 0.996 | 0.772 | 0.870 ±(0.112) |
| 58 | 7 | 0.992 | 0.826 | 0.901±(0.083) | 0.988 | 0.820 | 0.896 ±(0.084) |
| | 11 | 0.994 | 0.783 | 0.876±(0.106) | 0.992 | 0.781 | 0.874 ±(0.106) |
| | 15 | 0.979 | 0.782 | 0.869±(0.099) | 0.976 | 0.778 | 0.866 ±(0.099) |
| | 19 | 1.000 | 0.799 | 0.888±(0.101) | 1.000 | 0.766 | 0.867 ±(0.117) |
| 71 | 7 | 1.000 | 0.896 | 0.945±(0.052) | 1.000 | 0.890 | 0.941 ±(0.055) |
| | 11 | 1.000 | 0.889 | 0.941±(0.056) | 1.000 | 0.875 | 0.933 ±(0.063) |
| | 15 | 0.978 | 0.790 | 0.874±(0.094) | 0.978 | 0.784 | 0.870 ±(0.097) |
| | 19 | 0.996 | 0.801 | 0.888±(0.098) | 0.996 | 0.781 | 0.875 ±(0.108) |
| | 23 | 0.997 | 0.799 | 0.887±(0.099) | 0.996 | 0.760 | 0.862 ±(0.118) |
| 175 | 7 | 0.978 | 0.947 | 0.962±(0.016) | 0.953 | 0.913 | 0.933 ±(0.020) |

ruins. Isso ocorre porque quanto maior o deslizamento da janela, réplicas que poderiam estar em um determinado intervalo podem ficar em janelas diferentes e não serão comparados.

Vale ressaltar também que nesse *dataset* a precisão não foi influenciado pelo tamanho da janela e deslocamento. Já o valor da revocação foi influenciado porque quanto maior a janela, mais exemplos havia para o treinamento da PG e conseguia aprender mais. A tendencia observada é que quanto mais aumenta o tamanho da janela normalmente se tem um resultado melhor.

Tendo em vista que o valor da *Dmax* do *dataset Synthetic* foi 162, decidiu-se fazer um teste utilizando uma janela de tamanho 175 e deslocamento 7, apenas para mostrar que se pode ter bons resultados se a análise da distância obtido pelas funções de similaridade for bem observada.

Na Tabela 6.7 são apresentados os resultados referentes ao tempo gasto na etapa de treinamento, a porcentagem em comparação ao *PG configuração padrão* e a quantidade de réplicas encontrada na no arquivo de testes.

Observando o resultado do tempo, a janela de tamanho 71 com deslocamento 7, realizou o processo com 22.4% do tempo comparado ao *PG configuração padrão*

**Tabela 6.7.** Resultados dos experimentos no *dataset Synthetic*

| **Resultados: *dataset Synthetic*** | | | | | |
|---|---|---|---|---|---|
| Configurações (*Dataset* com 500 Registros) | | Tempo do Treino (Segundos) | | Total de Comparações | Réplicas Identificadas |
| (Total de réplicas no *dataset*: 171) | | Total | %Tempo | Qtd | Qtd |
| PG configuração padrão | | 2.179.795 | 100% | 124.750 | 169 |
| Tam. janela | Incremento | | | | |
| 12 | 7 | 70.182 | 03.2% | 3.935 | 107 |
| 23 | 7 | 171.810 | 07.9% | 9.313 | 114 |
| | 11 | 144.153 | 06.6% | 8.306 | 125 |
| 41 | 7 | 295.165 | 13.5% | 17.689 | 134 |
| | 11 | 299.020 | 13.7% | 16.635 | 129 |
| | 15 | 269.160 | 12.3% | 15.696 | 141 |
| 58 | 7 | 408.110 | 18.7% | 25.518 | 138 |
| | 11 | 426.960 | 19.6% | 24.580 | 137 |
| | 15 | 396.215 | 18.2% | 23.446 | 136 |
| | 19 | 377.575 | 17.3% | 22.668 | 131 |
| 71 | 7 | 488.785 | 22.4% | 31.158 | 154 |
| | 11 | 598.130 | 27.4% | 30.370 | 152 |
| | 15 | 423.240 | 19.4% | 29.001 | 148 |
| | 19 | 490.075 | 22.5% | 28.035 | 144 |
| | 23 | 519.250 | 23.8% | 26.959 | 141 |
| 175 | 7 | 1.078.600 | 49.5% | 70.455 | 168 |

encontrando mais réplicas do que todas as outras configurações. Já a janela com tamanho 71 com deslocamento 11 obteve resultado similar, porém realizou com tempo superior a 5% em relação a janela de tamanho 71 com deslocamento 7.

Uma das prováveis razões do PG não ter encontrado todas as réplicas seja porque talvez a função utilizada para fazer o agrupamento do registro não foi eficiente em colocar os registros similares próximos um dos outros.

### 6.4.3 Experimentos com o *dataset Cora*

Neste *dataset* a função que melhor agrupou registros similares foi a função Bigrama (Bigram [Carvalho et al., 2008a]).

Na Tabela 6.8 abaixo, são descritos os resultados obtidos pela técnica proposta tanto na etapa de treino quanto na etapa de teste.

Semelhantemente aos resultados dos *datasets* avaliados anteriormente, os melhores resultados obtidos no *dataset Cora* foram a partir das janelas de tamanho 58 e 71 com valor de deslocamento 7, tanto na revocação quanto no treino.

Tendo em vista que o valor da *Dmax* do *dataset Cora* foi 97, decidiu-se fazer um teste utilizando uma janela de tamanho 105 e deslocamento 7, apenas para mos-

**Tabela 6.8.** Resultados dos experimentos no *dataset Cora*

| **Resultados: *dataset Cora*** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Configurações Gerais | | Treino | | | Teste | | |
| Parâmetros | | Precisão | Revocação | F1 ($\sigma$) | Precisão | Revocação | F1 ($\sigma$) |
| PG configuração padrão | | 0.897 | 0.917 | 0.907±(0.010) | 0.901 | 0.892 | 0.896 ±(0.016) |
| Tam. janela | Deslocamento | | | | | | |
| 12 | 7 | 0.510 | 0.587 | 0.546±(0.039) | 0.500 | 0.583 | 0.538 ±(0.042) |
| 23 | 7 | 0.609 | 0.814 | 0.697±(0.103) | 0.575 | 0.811 | 0.673 ±(0.118) |
| | 11 | 0.602 | 0.723 | 0.657±(0.061) | 0.580 | 0.723 | 0.644 ±(0.072) |
| 41 | 7 | 0.890 | 0.795 | 0.840±(0.048) | 0.954 | 0.735 | 0.830 ±(0.110) |
| | 11 | 0.660 | 0.792 | 0.720±(0.066) | 0.635 | 0.785 | 0.702 ±(0.075) |
| | 15 | 0.712 | 0.870 | 0.783±(0.079) | 0.700 | 0.862 | 0.773 ±(0.081) |
| 58 | 7 | 0.970 | 0.758 | 0.851±(0.106) | 0.970 | 0.750 | 0.846 ±(0.110) |
| | 11 | 0.906 | 0.803 | 0.851±(0.051) | 0.873 | 0.755 | 0.810 ±(0.059) |
| | 15 | 0.710 | 0.872 | 0.783±(0.081) | 0.702 | 0.809 | 0.752 ±(0.054) |
| | 19 | 0.739 | 0.865 | 0.797±(0.063) | 0.726 | 0.859 | 0.787 ±(0.067) |
| 71 | 7 | 0.864 | 0.901 | 0.882±(0.019) | 0.859 | 0.897 | 0.878 ±(0.019) |
| | 11 | 0.855 | 0.824 | 0.839±(0.016) | 0.846 | 0.812 | 0.829 ±(0.017) |
| | 15 | 0.639 | 0.821 | 0.719±(0.091) | 0.613 | 0.783 | 0.688 ±(0.085) |
| | 19 | 0.702 | 0.794 | 0.745±(0.046) | 0.700 | 0.766 | 0.732 ±(0.033) |
| | 23 | 0.745 | 0.802 | 0.772±(0.029) | 0.734 | 0.796 | 0.694 ±(0.031) |
| 105 | 7 | 0.902 | 0.841 | 0.870±(0.031) | 0.925 | 0.792 | 0.853 ±(0.067) |

trar que pode-se ter bons resultados se a análise da distância obtido pelas funções de similaridade for bem observada.

Na Tabela 6.9 são apresentados os resultados referentes ao tempo gasto na etapa de treinamento, a porcentagem em comparação ao *PG configuração padrão* e a quantidade de réplicas encontradas.

Observando o resultado do tempo, a janela de tamanho 71 com deslocamento 7, realizou o processo com 33.1% do tempo comparado ao *PG configuração padrão* encontrando mais réplicas do que todas as outras configurações. Já a janela com tamanho 71 com deslocamento 11 obteve o mesmo resultado,e com tempo similar em relação a janela de tamanho 71 com deslocamento 7.

Da mesma forma que ocorreu no *dataset Synthetic*, uma das prováveis razões do PG não ter encontrado todas as réplicas seja porque talvez a função utilizada para fazer o agrupamento do registro não foi eficiente em colocar os registros similares próximos um dos outros.

Como citado anteriormente, o *dataset Cora* possui *strings* grandes para comparação tornando o tempo de treino maior do que nos outros *datasets* utilizadas em nossos experimentos.

**Tabela 6.9.** Resultados dos experimentos no *dataset Cora*

| **Resultados: *dataset Cora*** | | | | | |
|---|---|---|---|---|---|
| Configurações (*Dataset* com 200 Registros) | | Tempo do Treino (Segundos) | | Total de Comparações | Réplicas Identificadas |
| (Total de réplicas no *dataset*: 264) | | Total | %Tempo | Qtd | Qtd |
| PG configuração padrão | | 20.743.020 | 100% | 19.900 | 256 |
| Tam. janela | Incremento | | | | |
| 12 | 7 | 910.440 | 4.4% | 1.527 | 77 |
| 23 | 7 | 2.182.480 | 10.5% | 3.594 | 123 |
| | 11 | 1.913.452 | 9.2% | 3.257 | 113 |
| 41 | 7 | 4.049.760 | 19.5% | 6.552 | 174 |
| | 11 | 4.305.432 | 20.8% | 6.240 | 161 |
| | 15 | 3.714.321 | 17.9% | 5.796 | 148 |
| 58 | 7 | 5.713.200 | 27.5% | 9.264 | 194 |
| | 11 | 5.943.245 | 28.7% | 8.564 | 192 |
| | 15 | 5.523.4310 | 26.6% | 8.446 | 175 |
| | 19 | 5.234.243 | 25.2% | 8.076 | 175 |
| 71 | 7 | 6.872.960 | 33.1% | 10.991 | 203 |
| | 11 | 6.924.440 | 31.9% | 10.410 | 203 |
| | 15 | 6.143.921 | 29.6% | 10.101 | 195 |
| | 19 | 5.934.287 | 28.6% | 9.491 | 190 |
| | 23 | 5.638.592 | 27.2% | 9.318 | 186 |
| 105 | 7 | 8.411.400 | 40.6% | 14.749 | 254 |

## 6.5 Considerações Finais do Capítulo

Conforme os resultados apresentados, pode-se concluir que a tendência é que quanto maior o tamanho da janela os resultados são melhores. Isso porque quanto maior a janela mais exemplos são utilizados para o treinamento da PG. Já em relação a velocidade da janela, ocorre ao contrário, pois quanto maior o tamanho do deslocamento, as réplicas (exemplos positivos) que poderiam estar em um determinado intervalo podem ficar em janelas diferentes e não serão comparados.

Outra observação importante a ser destacada é que uma das razões prováveis do PG não ter encontrado todas as réplicas em alguns *datasets*, talvez seja porque a função utilizada para fazer o agrupamento do registro não foi eficiente em colocar os registros similares próximos um dos outros.

A desvantagem desse método é que dependendo do tamanho da janela deslizante, perde sempre a revocação. Isso porque muitos registros ficam fora do intervalo referente ao tamanho da janela deslizante e não são comparados. Porém, se a configuração for bem observada, os resultados podem ser eficientes e o tempo para processamento reduzido consideravelmente.

De modo geral, a partir da descrição experimental apresentada neste capítulo,

pode-se concluir que é possível tornar a etapa de treino da técnica de PG para deduplicação de registros mais rápida mantendo o nível de qualidade das soluções geradas. Nos experimentos, a técnica proposta foi capaz de obter uma eficácia próxima a abordagem de [Carvalho et al., 2008a], porém com uma redução significativa no tempo de treinamento.

Além disso, apesar das diferentes características de cada *dataset*, a técnica proposta apresenta resultados satisfatórios mostrando pode ser aplicado na maioria dos *datasets* onde utilize abordagens baseadas em PG para deduplicação de registros.

# Capítulo 7

# Conclusões e Trabalhos Futuros

Neste capítulo, são apresentadas as conclusões obtidas durante a realização desta dissertação, além da proposta para trabalhos futuros. Algumas conclusões foram obtidas durante a implementação da técnica, e outras puderam ser observadas já na fase de estudos preliminar.

## 7.1 Conclusões

Este trabalho propôs um método baseado na combinação de uma técnica de agrupamento e janela deslizante para a etapa de treinamento em programação genética. Onde a partir da seleção de um registro referência, todos os outros registros serão comparados a ele por uma função de similaridade pré estabelecida, normalmente específica por base. Com isso, os registros serão agrupados com base nas distâncias retornadas pela função de similaridade. Em seguida, serão comparados de acordo com o tamanho definido em uma estratégia de janela deslizante com incremento, também proposta nesta dissertação.

A nossa combinação das técnicas, foi testada em três conjuntos de dados e comparadas com os resultados obtidos por [Carvalho et al., 2008a] (utilizado como *baseline*). Os resultados reforçaram a ideia de que a utilização dessa técnica permite a obtenção de resultados satisfatórios, com a vantagem da redução considerável tempo de treinamento da PG, além de manter a qualidade dos resultados. Os experimentos demonstraram que utilizar o tamanho da janela deslizante próximo ou acima a distância máxima dos registros ordenados pela similaridade *Dmax*, obtém os melhores resultados. A redução obtida no tempo de experimento total entre 50% a 70% comparado ao tempo da abordagem utilizada como *baseline* para os testes.

## 7.2 Trabalhos Futuros

A partir dos resultados apresentados, diversos trabalhos podem ser realizados para complementar a abordagem proposta nesta dissertação. Pode-se destacar como sugestões de trabalhos futuros, os seguintes pontos:

1) Em relação a parte de agrupamento:

   - Pretende-se fazer experimentos alterando a função de comparação entre os registros, ou seja, pode-se fazer uma combinação entre várias funções.
   - Utilizar alguma técnica que gere uma função que maximize a proximidade entre os registros originais e réplicas.

2) Em relação a estratégia de janela deslizante:

   - Automatizar o processo de definição da função de similaridade (neste trabalho é feita manualmente), ou seja, carregar a base experimental e testar todas funções de similaridade procurando a menor entre as maiores distâncias obtidas para cada função, e escolher a melhor função, para em seguida realizar o restante das etapas.
   - Utilizar paralelismo para a utilização da estratégia de janela deslizante.
   - Utilizar Map-Reduce[1] para paralelizar todo o processo em larga escala com a estratégia de janela deslizante.
   - Realizar um estudo experimental utilizando outros repositórios de dados reais com diferentes domínios e graus de dificuldade, para ajudar a consolidar e estender os resultados obtidos neste trabalho.
   - Aplicar a técnica proposta juntamente com outras propostas que visam reduzir o tempo de treino de PG, como por exemplo, a abordagem de [Gonçalves, 2009], que utiliza uma técnica determinística para sugerir automaticamente exemplos de treino. Isso pode reduzir ainda mais o tempo de treinamento.

[1]Map-Reduce é um paradigma de programação introduzido pelo Google para processar e analisar grandes conjuntos de dados

# Referências Bibliográficas


[Ankerst et al., 1999] Ankerst, M. et al. (1999). OPTICS: Ordering Points to Identify the Clustering Structure. In: Proceedings of the ACM SIGMOD Conference on Management of Data, p. 49-60, Philadelphia, PA, USA, June, 1999.

[Baeza-Yates, 2011] Baeza-Yates, R.; Ribeiro-Neto, B. (2011). Modern Information Retrieval: The Concepts and Technology Behind Search. *ACM Press / Addison-Wesley.*, 2nd.ed. [S.l.].

[Banzhaf et al., 1998] Banzhaf, W. et al. (1998). Genetic Programming: An Introduction: On The Automatic Evolution of Computer Programs and Its Applications. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.*

[Bhattacharya, 2004] Bhattacharya, I. & Getoor, L. (2004). Iterative record linkage for cleaning and integration. *In Proceeding of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.*, p. 11-18, Paris, France.

[Bilenko et al., 2006] Bilenko, M. et al. (2006). Adaptive Blocking: Learning to Scale up Record Linkage. *In Proceedings of the 6th IEEE International Conference on Data Mining.*, p. 87-96. ICDM2006, IEEE, Inc.

[Bilenko, 2003] Bilenko, M.; Mooney, R. J. (2003). Adaptive Duplicate Detection Using Learnable String Similarity Measures. *In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, New York, NY, USA. p. 39-48. (KDD 03).

[Bishop, 2006] Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer, ISBN: 0387310738.*

[Carvalho et al., 2006] Carvalho, M. G. et al. (2006). Learning to deduplicate. *In: ACM IEEE Joint Conference on Digital Libraries.*, p. 41-50.

[Carvalho et al., 2008a] Carvalho, M. G. et al. (2008a). The impact of parameter setup on a genetic programming approach to record deduplication. *Sociedade Brasileira de Computação, In: Brazilian Symposium on Databases*, p. 91-105.

[Carvalho et al., 2008b] Carvalho, M. G. et al. (2008b). Replica identification using genetic programming. *In: ACM Symposium on Applied Computing.*, p. 1801-1806.

[Carvalho et al., 2012] Carvalho, M. G. et al. (2012). A genetic programming approach to record deduplication. *IEEE Transactions on Knowledge and Data Engineering, Piscataway, NJ, USA.*, v.24, n.3, p. 399-412.

[Chaudhuri et al., 2003] Chaudhuri, S. et al. (2003). Robust and Efficient Fuzzy Match for Online Data Cleaning. *In: ACM SIGMOD International Conference on Managment of Data Conference.*, [S.l.: s.n.], p. 313-324.

[Christen, 2007] Christen, P. (2007). Towards Parameter-Free Blocking for Scalable Record Linkage. *ANU Joint Computer Science Technical Report Series.*, Agosto/2007.

[Christen, 2012] Christen, P. (2012). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Eng.*, 24(9) : 1537-1555.

[Cohen, 2002] Cohen, W. W.; Richman, J. (2002). Learning to match and cluster large high-dimensional data sets for data integration. *In Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, S.l.: s.n., p. 475-480.

[Dal Bianco et al., 2011] Dal Bianco, G. et al. (2011). A fast approach for parallel deduplication on multicore processors. *In: ACM Symposium on Applied Computing.*, New York, NY, USA. Proceedings... p.1027-1032.

[Dal Bianco et al., 2013] Dal Bianco, G. et al. (2013). Tuning large scale deduplication with reduced effort. *In Proceedings International Conference on Scientific and Statistical Database Management, ACM, new york.*, p. 18:1-18:12. (SSDBM).

[Dal Bianco et al., 2015] Dal Bianco, G. et al. (2015). A practical and effective sampling selection strategy for large scale deduplication. *In: IEEE Transactions on Knowledge and Data Engineering 27(9) : 1-1.*

[Elmagarmid et al., 2007] Elmagarmid, A. K. et al. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, p. 1-16.

[Elmasri, 2013] Elmasri, R., N. S. (2013). Sistemas de Bancos de Dados. 6. ed. São Paulo: Addison Wesley.

[Evangelista et al., 2009] Evangelista, L. et al. (2009). Blocagem adaptativa e flexível para o pareamento aproximado de registros. XXIV Simpósio Brasileiro de Banco de Dados.

[Fellegi, 1969] Fellegi, I. P; Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association.*, [S.l.], v.64, n.328, p. 1183-1210.

[Freitas et al., 2010] Freitas, J. et al. (2010). Active learning genetic programming for record deduplication. *In Proceedings IEEE Congress on Evolutionary Computation.*, [S.l.: s.n., p. 1-8.

[Gantz, 2012] Gantz, John; Reinsel, D. (2012). Extracting Value from Chaos. Disponível em: http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf. [Acessado em: 12-11-2015.].

[Goldberg, 1989] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. *Addison-Wesley Longman Publishing Co., Inc., New York, NY, USA.*

[Gonçalves, 2009] Gonçalves, G. e. a. (2009). Seleção Automática Exemplos de Treino para um Método de Deduplicação de Registros Baseado em Programação Genética. *In Anais do XXIV Simpósio Brasileiro de Bancos de Dados.*

[Guha et al., 2004] Guha, s. et al. (2004). Merging the Results of Approximatematch Operations. International Conference on Very Large Data Bases, VLDB, 30., 2004. p. 636-647.

[Holland, 1975] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor, MI, USA.*

[Jain et al., 1999] Jain, A. K. et al. (1999). Data clustering: A review. *ACM Computing Surveys. 31(3)8.*

[Koudas et al., 2006] Koudas, N. et al. (2006). Record linkage: Similarity measures and algorithms. *In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data.*, p. 802-803, Chicago, IL, USA.

[Koza, 1992] Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. *The MIT Press, Cambridge, MA.*

[Linden, 2012] Linden, R. (2012). Algoritmos Genéticos. 3 ed. Rio de Janeiro: Ciência Moderna Ltda, p. 484.

[Manning et al., 2008] Manning, C. D, . et al. (2008). Introduction to information retrieval. *New York, NY, USA: Cambridge University Press.*

[Mccallum et al., 2000] Mccallum, A. et al. (2000). Efficient Clustering of High-Dimensional Data Sets With Application to Reference Matching. *In: ACM Knowledge Discovery and Data Mining.*, Anais ACM SIGKDD.

[Michelson, 2006] Michelson, M.; Knoblock, C. A. (2006). Learning Blocking Schemes for Record Linkage. *In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI).*

[Mitchell, 2007] Mitchell, T. M. (2007). Machine Learning. McGraw-Hill Science - Engineering - Math, ISBN 0070428077.

[Newcombe et al., 1959] Newcombe, H. B. et al. (1959). Automatic Linkage of Vital Records. v. 130, n. 3381, p. 954 - 959.

[Santos et al., 2008] Santos, J. B. et al. (2008). Automatic Threshold Estimation for Data Matching Applications. *Sociedade Brasileira de Computação, In: Brazilian Symposium on Databases, 23*, p. 106-119. (SBBD 08).

[Silberschatz et al., 2012] Silberschatz, A. et al. (2012). Sistema de Banco de Dados. 6. ed. Elsevier - Campus. ISBN - 9788535245356.

[Tejada et al., 2001] Tejada, S. et al. (2001). Learning Object Identification Rules for Information Integration. *Inf. Syst.*, p. 607-633.

[Tejada et al., 2002] Tejada, S. et al. (2002). Learning domain-independent string transformation weights for high accuracy object identification. *In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, AB, Canada.*, p. 350-359.

[Vernica et al., 2010] Vernica, R. et al. (2010). Efficient parallel set-similarity joins using MapReduce. *In: Proceedings of the 2010 International Conference on Management of Data.*, New York, NY, USA. Anais. . . ACM, 2010. p.495-506.

[Verykios et al., 2003] Verykios, V. S. et al. (2003). A bayesian decision model for cost optimal record matching. *The VLDB Journal.*, 12(1):28-40.

[Whang, 2012] Whang, S.; Garcia-Molina, H. (2012). Joint Entity Resolution. *IEEE International Conference on Engineering.*, [S.l.: s.n.], p. 294-305.

[Whang, 2013] Whang, S.; Garcia-Molina, H. (2013). Joint Entity Resolution on Multiple Datasets. *The VLDB Journal.*, [S.l.], v.22, n.6, p. 773-795.

[Winkler, 1999] Winkler, W. E. (1999). The State of Record Linkage and Current Research Problems. *Technical report, Statistical Research Division.*

[Woolf, 2010] Woolf, B. (2010). Facebook Statistics. Kissmetrics, Analytics Built to Optimize Marketing. Group & segment to find more of your best customers. Disponível em: http://kissmetrics.com/facebook-statistics. [Acessado em: 12-11-2015.].

[Zaufman, 1990] Zaufman, L.; Rousseeuw, P. J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis: Jonh Wiley & Sons.

[Zhang et al., 2001] Zhang, B. et al. (2001). K-Harmonic Means - A Spatial Clustering Algorithm With Bossting. *In: RODDICK, J.F.; HORNSBY, K., Eds., Temporal, Spatial and Spatio-Temporal Data Mining, Lecture Notes in Artificial Intelligence.*, Berlin: Springer, p. 31-45.

[Ziv et al., 1977] Ziv, J. et al. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3) pp. 337-343.