UNIVERSIDADE ESTADUAL DE MONTES CLAROS
Centro de ciências Exatas e Tecnológicas
Programa de Pós Graduação em Modelagem Computacional e Sistemas

Herberth Giuliano Amaral Silva

CONSTRUÇÃO INDUTIVA EVOLUCIONÁRIA APLICADA AO PROBLEMA DE DEDUPLICAÇÃO DE REGISTROS

Montes Claros - MG

Outubro de 2017

Herberth Giuliano Amaral Silva

CONSTRUÇÃO INDUTIVA EVOLUCIONÁRIA APLICADA AO PROBLEMA DE DEDUPLICAÇÃO DE REGISTROS

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional e Sistemas do Instituto de Ciências Exatas da Universidade Estadual de Montes Claros como requisito parcial para a obtenção do grau de Mestre em Modelagem Computacional e Sistemas

Orientador: **Prof. Dr. Renê Rodrigues Veloso**
Coorientador: **Prof. Dr. João Batista Mendes**

Montes Claros - MG
Outubro de 2017

S586c

Silva, Herberth Giuliano Amaral.
Construção indutiva evolucionária aplicada ao problema de deduplicação de registros [manuscrito] / Herberth Giuliano Amaral Silva. – 2017.
48 f. : il.

Bibliografia: f. 34-36.
Dissertação (mestrado) - Universidade Estadual de Montes Claros - Unimontes, Programa de Pós-Graduação em Modelagem Computacional e Sistemas/PPGMCS, 2017.

Orientador: Prof. Dr. Renê Rodrigues Veloso.
Coorientador: Prof. Dr. João Batista Mendes.

1. Deduplicação de registros. 2. Identificação de duplicatas. 3. Genética - Programação. 4. Indução de árvores de decisão. I. Veloso, Renê Rodrigues. II. Mendes, João Batista. III. Universidade Estadual de Montes Claros. IV. Título.

RELATÓRIO DE DEFESA DE DISSERTAÇÃO

MESTRADO

Universidade Estadual de Montes Claros - Unimontes
Pró-Reitoria de Ensino/Coordenadoria de Pós-Graduação
Programa de Pós-Graduação em Modelagem Computacional e Sistemas - PPGMCS

1 - Identificação do Aluno

Nome: Herberth Giuliano Amaral Silva	Matrícula: 180265
Linha de Pesquisa: Inteligência Computacional, Otimização e suas Aplicações.	

2 - Sessão de Qualificação

Título: "**Construção Indutiva Evolucionária Aplicada ao Problema de Deduplicação de Registros**"

3 - Comissão Examinadora

Nome	Função	Assinatura
Prof. Dr. Renê Rodrigues Veloso - UNIMONTES	Orientador(a)	
Prof. Dr. João Batista Mendes - UNIMONTES	Co-orientador(a)	
Prof. Dr. Maurílio José Inácio - UNIMONTES	Examinador(a)	
Prof. Dr. Honovan Paz Rocha (IECT/UFVJM)	Examinador(a)	

4 - Resultado

A comissão Examinadora, em **31/10/2017** após Defesa de Dissertação e arguição do (a) candidato(a), decidiu:

( ) pela aprovação da Dissertação
( ) pela reprovação da Dissertação
(X) pela revisão de forma, indicando o prazo de 30 dias para apresentação definitiva.
( ) pela reformulação da Dissertação, indicando o prazo de _____ dias para nova versão.

Preencher somente em caso de revisão de forma:	**Preencher somente em caso de revisão de reformulação:**
(X) O(a) aluno(a) apresentou a revisão de forma e a Dissertação foi aprovada.	( ) O(a) aluno(a) apresentou a reformulação e a Dissertação foi aprovada.
( ) O(a) aluno(a) apresentou a revisão de forma e a Dissertação foi reprovada.	( ) O(a) aluno(a) apresentou a reformulação e a Dissertação foi reprovada.
( ) O(a) aluno(a) não apresentou a revisão da forma.	( ) O(a) aluno(a) não apresentou a reformulação.

Autenticação
Orientador(a) Comissão Examinadora

01/12/2017 _____
Data Assinatura

Autenticação
Coordenador

01/12/2017 _____
Data Assinatura

À minha noiva, Luciana Balieiro, pelo incentivo, companheirismo e compreensão durante esta e outras jornadas.

Agradecimentos

Durante os quase três últimos anos, várias pessoas foram essenciais para a realização deste trabalho.

Quero inicialmente agradecer ao meu orientador, Prof. Dr. Renê Rodrigues Veloso, por aceitar minha teimosia e me orientar em um tema que não é sua especialidade e me mostrar que eu não devo temer notações matemáticas mais complexas.

Ao meu coorientador, Dr. João Batista Mendes, que teve a paciência de me auxiliar mesmo em uma área que, apesar de similar, não é a dele.

Meus sinceros agradecimentos aos professores Dr. Marcos Flávio Silveira Vasconcelos D'Angelo e Dr. Nilton Alves Maia, por dedicarem esforços na abertura e manutenção do Programa de Pós-Graduação em Modelagem Computacional e Sistemas (PPGMCS). Sem esse trabalho, essa tão importante etapa não seria sequer possível.

Aos meus avós, Iracema Gomes Amaral e José Marcos Amaral, que sempre me cobraram a busca pelo conhecimento e sabedoria.

Aos meus pais, José Osmar da Silva e Rosimary Gomes Amaral Silva, pelo apoio e compreensão nas ausências das reuniões de família, sobre o meu pretexto de dedicação ao mestrado.

Aos meus amigos Everton Fernandes, Luana Balieiro, Gabriel Oliveira, Edmar Ferreira, Victor Hugo e Steve Lacerda, por aceitarem as desculpas esfarrapadas de ausência em todo esse período.

E finalmente, os meus profundos agradecimentos à minha noiva Luciana Balieiro, por insistir tanto, e apoiar na mesma proporção, no início, meio e fim deste curso.

Resumo

Identificar corretamente registros duplicados em grandes bases de dados é uma tarefa complexa e necessária para manter a consistência de bases de dados. Em um hospital, por exemplo, ter registros duplicados de pacientes pode gerar desde pequenas inconveniências administrativas a erros de diagnósticos devido à divisão não-intencional dos dados entre os registros.

O processo de deduplicação de registros pode ser considerado complexo por envolver vários estágios e técnicas e é pesquisado desde a década de 1960, tendo início com o trabalho de (FELLEGI; SUNTER, 1969). Desde então, diversas abordagens têm sido apresentadas na literatura buscando-se melhorar a eficácia desse tipo de tarefa, como a abordagem descrita por (CARVALHO et al., 2012), que, até onde vai nosso conhecimento, foi o primeiro a aplicar programação genética no processo de deduplicação de registros.

Neste trabalho, apresenta-se uma abordagem de aprendizado supervisionado baseada em programação genética (GP) e árvores de decisão, gerando classificadores capazes de identificar duplicatas de registros com altos índices de precisão e revocação.

Para tanto, é apresentada uma nova heurística de geração de população inicial do GP, além da definição de regras de ligação utilizando uma abordagem baseada em pontos de corte variáveis. Também é apresentado o uso de construção indutiva evolucionária, que utiliza o melhoramento dos indivíduos criado pelo GP para gerar novas propriedades na base de dados e classificá-las posteriormente utilizando indução de árvores de decisão.

Os resultados experimentais mostram que a abordagem apresenta melhorias em termos de expressividade e de *fitness* das soluções geradas, mesmo em seu caso mais simples, quando comparados ao estado-da-arte.

Palavras-Chave: Deduplicação de registros, Identificação de duplicatas, programação genética, indução de árvores de decisão.

Abstract

The correct identification of duplicate records in large databases is a complex and necessary task to keep databases consistency. In a hospital environment, for instance, having duplicate patient records can cause troubles as small as simple administrative inconveniences through errors of diagnose due to non-intentional data split between records.

The record deduplication process can be complex because its many stages and techniques, and it has been researched since 1960s starting with (FELLEGI; SUNTER, 1969). Since then, several approaches have been presented on the literature aiming to improve the efficiency of this kind of task, just like the approach described by (CARVALHO et al., 2012), which, to the best of our knowledge, was the first one to use genetic programming in record deduplication processes.

We present a supervised learning based approach, which is based on genetic programming (GP) and decision trees that generates classifiers capable of record duplication identification with high precision and recall levels.

Therefore, we present a new genetic programming initial population generation heuristic and a new linkage rule definition with a variable cut-point based approach.

The experimental results show that the presented approach improves solutions' expressivity and fitness even in simplest cases when compared to the state-of-the-art.

Keywords Record deduplication, Record linkage, Genetic programming, Decision tree induction.

Lista de ilustrações

Lista de tabelas

Acrônimos

1. **GP** - Programação Genética (*Genetic Programming*);
2. **GA** - Algoritmo Genético (*Genetic Algorithm*);
3. **ECI** - Construção Indutiva Evolucionária (*Evolutionary Constructive Induction*);
4. **SVM** - Máquina de vetores de suporte (*Support Vector Machine*);
5. **PC** - Ponto de corte.

Sumário

1 Introdução

Deduplicação de registros[1] é a tarefa de identificar, em um repositório de dados, registros diferentes mas que referem-se a uma mesma entidade, na qual essa diferença ocorre devido a ruídos nos dados (como erros de escrita, diferente representação de dados, informações faltantes, falta de padronização ou uma combinação desses fatores (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007)). Como um exemplo prático, pode-se citar os registros de pacientes duplicados dentro de uma ou mais bases de dados de sistemas de informações da área da saúde. Trata-se de um problema de fundamental importância, uma vez que a duplicação de registros pode prejudicar severamente os resultados de qualquer processamento ou mineração de dados subsequente (CHRISTEN, 2012).

O problema de deduplicação de registros tem sido abordado por diversos trabalhos, entre eles (CARVALHO et al., 2012), que foi um dos primeiros a utilizar programação genética para evoluir programas capazes de detectar registros duplicados. (ISELE; BIZER, 2012) aprimora a abordagem de (CARVALHO et al., 2012) com o uso de técnicas que melhoram a expressividade das regras de ligação construídas. Técnicas que combinam programação genética com aprendizado ativo (*active learning*) são bastante exploradas em (FREITAS et al., 2010), (NGOMO; LYKO, 2012), (SUN et al., 2017), (ISELE; BIZER, 2013). Este trabalho não discutirá as abordagens baseadas em aprendizado ativo por possuírem uma natureza diferente da daqui apresentada.

De forma mais específica, o problema de deduplicação de registros busca obter dois conjuntos disjuntos M e U, a partir de uma base de dados $A = \{r_1, r_2, \ldots, r_n\}$, onde r_i é uma tupla com $P(A)$ atributos ou propriedades, de forma que r_i^p, com $p \in P(A)$, é um item da base de dados. Sejam os conjuntos $U, M \subset A^2$, sendo que M (*matched*) contêm os pares de registros diferentes que correspondem a mesma entidade e U (*unmatched*) contêm os pares de registros diferentes que não correspondem a mesma entidade. Por exemplo, considere a Tabela 1, o conjunto $P(I) = \{id, nome, sobrenome, num\}$. Nessa tabela, o registro da segunda linha é referenciado por r_2 e o item "lily" é acessado por r_2^{nome}. Neste exemplo, o conjunto $M = \{(r_2, r_5), (r_3, r_4)\}$ foi obtido por evidências sobre similaridades entre as propriedades *nome* e *sobrenome*. O conjunto U contêm todos os pares que não estão em M.

[1] Também conhecido por limpeza de dados (CARVALHO et al., 2012), *record linkage* (CARVALHO et al., 2012) (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007) (FELLEGI; SUNTER, 1969), detecção de duplicatas (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007), *data scrubbing* (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007), *record matching* (CARVALHO et al., 2012), identificação de instâncias, *name matching*, *database hardening*, resolução de correferência e incerteza de identidade (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007).

Tabela 1 – Exemplo de base de dados para deduplicação de registros

id.	nome	sobrenome	num
1	Lily	Quinton	2 Baker St.
2	lily	braddy	137
3	conne	Kampbeol	49
4	Connor	Cambell	059
5	Lily	Braddy	no. 2
6	CONOR QUINTON		78

1.1 Objetivos

O objetivo deste trabalho é a busca de formas de geração de regras de ligação mais eficientes do ponto de vista de eficácia (*fitness*) para deduplicação de registros utilizando programação genética e indução de árvores de decisão. É sabido, através do trabalho desenvolvido em (CARVALHO et al., 2012), que o uso da programação genética é um método eficaz para construir regras de ligação. De forma mais específica, este trabalho busca:

- Desenvolver técnicas para diminuir o tamanho dos indivíduos solução gerados via programação genética, aumentando assim sua expressividade sem diminuir o poder de classificação;
- Explorar métodos definidos na literatura para tornar a escolha do ponto de corte mais eficiente;
- Investigar a viabilidade de combinar os resultados obtidos até então com a programação genética com árvores de decisão através da técnica de Construção Indutiva Evolucionária (MUHARRAM; SMITH, 2005).

1.2 Abordagem utilizada

Para atingir os objetivos deste trabalho, a abordagem adotada baseia-se na abordagem de (CARVALHO et al., 2012), que utiliza indivíduos gerados via programação genética para construção de classificadores com as seguintes diferenças:

1. Emprego de uma nova metodologia de cálculo de ponto de corte: como os indivíduos são expressões matemáticas que produzem um valor numérico, é necessário um ponto de corte para classificação. Esse ponto de corte possui um valor fixo em (CARVALHO et al., 2012). Este trabalho emprega uma nova metodologia de cálculo de ponto de corte baseado no conceito de ganho de informação (utilizado para treinar árvores de decisão) para calcular um ponto de corte específico para cada indivíduo.

Como consequência, o indivíduo gerado via programação genética não contem nós com valores absolutos para balanceamento;

2. Uso de uma nova abordagem para geração de população inicial, que constrói indivíduos com o menor número de elementos possível de forma semi aleatória;

3. Combinação de programação genética com árvores de decisão para melhoria do poder de classificação.

1.3 Contribuições

As contribuições obtidas com o emprego da abordagem utilizada podem ser resumido da seguinte forma:

- Uma abordagem mais eficiente baseada no conceito de ganho de informação para determinar melhores pontos de corte para os classificadores baseados em programação genética (*Genetic Programming* - GP), em contraste com a abordagem descrita em (CARVALHO et al., 2012) de uso de um ponto de corte fixo;

- Uma nova heurística de geração de população inicial da GP. Essa heurística tem como característica principal a geração semi aleatória dos menores indivíduos possíveis;

- O uso da GP para descobrir as melhores combinações de evidências e posterior uso dessas evidências por uma árvore de decisão para classificação.

Os resultados obtidos com a aplicação da abordagem utilizada igualam-se ou superam os resultados obtidos por (CARVALHO et al., 2012), mesmo quando considerado aplicações parciais da abordagem descrita. Além dos melhores resultados, os indivíduos (soluções do problema) gerados através do método proposto neste trabalho são mais simples (ou seja, possuem menos nós) que os descritos por (CARVALHO et al., 2012).

1.4 Organização do texto

A redação do presente texto está organizada da seguinte forma: o Capítulo 2 tem como objetivo fundamentar o problema e a abordagem utilizada é descrita no Capítulo 3. O Capítulo 4 apresenta e discute os experimentos e resultados feitos com a abordagem descrita no Capítulo 3 e o Capítulo 5 apresenta as considerações finais.

2 Fundamentação Teórica

2.1 O problema de deduplicação de registros

O objetivo do processo de deduplicação de registros é encontrar um classificador capaz de associar um par qualquer de registros (r_a, r_b), com $a \neq b$, aos conjuntos M (mesmo registro) ou U (registros diferentes). No entanto, a classificação dos pares é apenas um dos passos do processo. Em geral, esse processo pode ser resumido com os passos seguintes (CHRISTEN, 2012), posteriormente ilustrado na Figura 1:

1. Limpeza e padronização das bases de dados;
2. Indexação das bases de dados. Este passo também é conhecido como *blocagem*, pois divide os grupos de comparação em blocos com o intuito de diminuir a quantidade de comparações feitas;
3. Comparação dos pares de registros. Essa comparação produz um *vetor de similaridades* V para todos os pares de tuplas;
4. Classificação do vetor de similaridades $v \in V$ de cada par de tuplas em duas (mesmo registro/registro diferente) ou três (mesmo registro/registro diferente/potencialmente o mesmo registro) classes diferentes;
5. Revisão manual dos pares que podem ser potencialmente o mesmo registro, caso a abordagem inclua a classe "potencialmente mesmo registro";
6. Avaliação do processo de classificação.

O processo apresentado é ilustrado pela Figura 1 e a descrição de cada passo é dada logo em seguida.

A necessidade do passo 1 reside na probabilidade de bases de dados reais conterem dados ruidosos, incompletos ou com formatação incorreta. Dentre possíveis ações de limpeza de dados encontram-se o uso de somente letras maiúsculas/minúsculas, a remoção de acentos e sinais, uso da versão fonética de nomes próprios e a padronização de contrações (ex., Sr. para Senhor, Av. para Avenida, entre outros). Por exemplo, a base de dados representada na Tabela 2 possui as mesmas informações da Tabela 1 (disponível no capítulo anterior), porém, padronizadas.

Com o intuito de diminuir a quantidade de comparações necessárias, o passo 2 é responsável por criar pares de registros candidatos à comparação mais detalhada, eliminando os pares que possuem pouca chance de serem realmente duplicados. Esse passo é

Figura 1 – Ilustração do processo de deduplicação de registros (CHRISTEN, 2012) (adaptado)

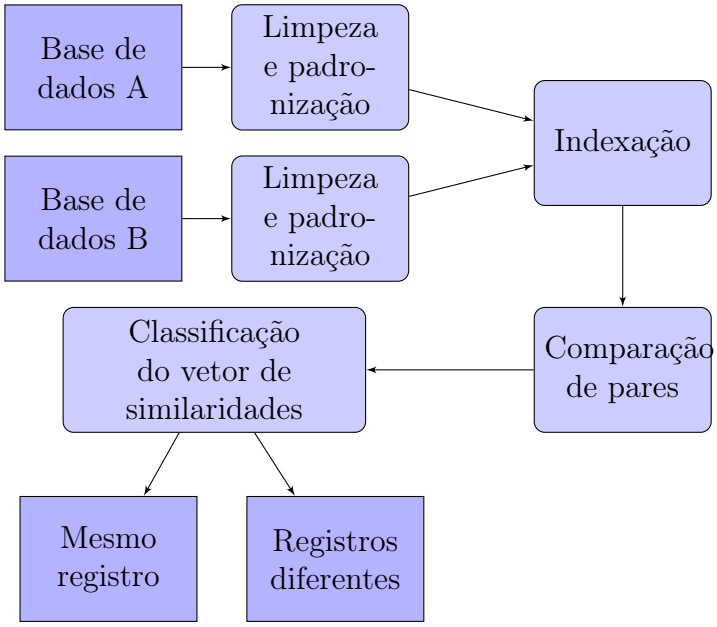


Tabela 2 – Exemplo de base de dados sem ruídos, baseada na Tabela 1.

id.	nome	sobrenome	num
1	lily	quinton	2
2	lily	braddy	137
3	conne	kampbeol	49
4	connor	cambell	59
5	lily	braddy	2
6	conor	quinton	78

necessário, pois no processo de deduplicação de uma base de dados A, $|A| * (|A| - 1)/2$ comparações precisam ser feitas (em que $|A|$ é a quantidade de registros da base de dados A), o que é assintoticamente quadrático (ou seja, $\mathcal{O}(n^2)$). Isso quer dizer que em uma base de dados com um milhão de registros, seria necessário realizar 10^{12} (um trilhão) de comparações.

Apesar de não diminuir a complexidade teórica do problema, as técnicas de indexação permitem que o espaço de comparação seja reduzido drasticamente [1], ao custo da diminuição da *revocação*[2], fazendo com que o número de comparações necessárias torne-se quase-linear à medida que o número de registros crescem linearmente. Apenas como exemplo, a lista (não exaustiva) a seguir mostra algumas das técnicas mais comuns de indexação/blocagem, em que somente os itens do mesmo bloco são comparados entre si:

- n **primeiros caracteres** - Técnica que coloca no mesmo bloco os pares de registros

[1] Mais de 98% segundo (CHRISTEN, 2012)

[2] Também conhecida como *recall*, é a possibilidade de encontrar todos os pares duplicados, não importando se falsos positivos foram encontrados no processo.

em que uma propriedade p qualquer tenha os mesmos n caracteres. Por exemplo, se consideramos os 2 primeiros caracteres, "Thiago" e "Thalis" ficariam no mesmo bloco e seriam comparados entre si;

- ***n-grams*** - Um $n-gram$[3] é o conjunto de tuplas de tamanho n construído contiguamente a partir de uma *string*. Por exemplo, o conjunto $3-gram$ da string "Thiago" é $\{('Thi','hia','iag','ago')\}$. Desta forma, pode-se colocar no mesmo bloco, todos os pares de registro que possuem pelo menos um *3-gram* em comum;

- **Indexação fonética** - Técnica de indexação que utiliza um algoritmo de representação fonética de *strings* com o intuito de desconsiderar pequenas variações de grafia. Por exemplo, "Thiago" e "Tiago" podem ser o mesmo registro com uma pequena variação de grafia, mas cuja representação fonética "tiagu" é a mesma. Neste caso, pode-se colocar todos os nomes foneticamente iguais no mesmo bloco de comparação;

- **Combinação de duas ou mais abordagens** - É possível, e muitas vezes até desejado, que mais de uma técnica de indexação seja utilizada em conjunto para conseguir melhores índices de revocação ou diminuir ainda mais a quantidade de comparações. Por exemplo, é possível combinar n primeiros caracteres com a indexação fonética para diminuir a quantidade de blocos ou para aumentar a revocação ao custo de mais pares comparados.

Em (CHRISTEN, 2012) há um trabalho de revisão abordando as principais técnicas de indexação/blocagem e suas respectivas avaliações em termos de revocação, precisão e diminuição da quantidade de comparações necessárias.

O passo 3 é responsável por transformar o conjunto de pares de registros A^2 em uma matriz numérica de vetores de similaridade V, que será utilizado posteriormente para classificação. Depois desse passo, diversos algoritmos de classificação podem ser utilizados: redes neurais artificiais, máquinas de vetores de suporte (*Support Vector Machines* - SVM), árvores de decisão, Naïve Bayes, dentre outros. Neste trabalho, usa-se classificadores baseados em programas gerados através do algoritmo de programação genética.

Formalmente, o passo 3 produz um conjunto de vetores de similaridade V (2.1) com os resultados de cada uma das comparações v_{ab} (2.2) entre os pares de registros $(r_a, r_b) \in A^2$. Cada item desse vetor de similaridades é formado pelas aplicações das *evidências* (CARVALHO et al., 2012).

$$V = \{v_{ab} \mid (r_a, r_b) \in A^2\} \tag{2.1}$$

[3] Também conhecido como *q-gram*.

$$v_{ab} = \{E_{<f,p>}(r_a, r_b) \mid f \in F_s, p \in P(A)\} \tag{2.2}$$

Em termos de classificação auxiliada por programas construídos automaticamente via programação genética, uma evidência $E_{<f,p>}$ é uma função definida como $E_{<f,p>} : A^2 \rightarrow \mathbb{R}$, sendo $E_{<f,p>}(r_a, r_b) = f(r_a^p, r_b^p)$, na qual $f \in F_s$ é uma das funções de similaridade aplicada aos itens r_a^p e r_b^p. As funções de similaridade, detalhadas na Tabela 3 tomam como entrada duas *strings* e retornam um valor numérico $d \in \mathbb{R}$ que representa o nível de similaridade das duas *strings*. A Tabela 4 mostra um conjunto de evidências utilizando as funções Levenshtein (L) e Jaro-Winkler (JW) com seus respectivos valores da base de dados da Tabela 2. Por exemplo, a evidência $E_{<nome,L>}$ compara dois registros utilizando a propriedade *nome* e a função de similaridade Levenshtein, na qual os valores próximos de 0 indicam itens semelhantes, ao passo que $E_{<nome,JW>}$ compara dois registros utilizando a propriedade nome com a função de similaridade Jaro-Winkler, na qual os valores próximos de 1 indicam itens semelhantes.

Tabela 3 – Conjunto F_s das funções de similaridade e suas características. Nas funções com relação "direta", quanto maior o valor, mais similar. Enquanto as funções com relação "inversa"quanto maior o valor, menos similar. $len(str)$ refere-se ao tamanho da maior *string* da comparação.

Função	Nome	Domínio	Relação
J	Jaro (JARO, 1989)	$0 \ldots 1$	Direta
JW	Jaro-Winkler (WINKLER, 1990)	$0 \ldots 1$	Direta
L	Levenshtein (LEVENSHTEIN, 1966)	$0 \ldots len(str)$	Inversa
Ng	N-grams (SHANNON, 1948)	$0 \ldots 1$	Direta
DL	Damerau-Levenshtein (DAMERAU, 1964)	$0 \ldots len(str)$	Inversa
DH	Distância de Hamming (HAMMING, 1950)	$0 \ldots len(str)$	Inversa
Cos	Similaridade de cossenos (TAHIR, 2015)	$-1 \ldots 1$	Direta

A classificação (passo 4) é responsável por tomar os valores das evidências (Tabela 4) e classificá-los em duas classes distintas: mesmos registros (M) e registros diferentes (U). Os conjuntos M e U são obtidos por meio da aplicação de uma regra de ligação $L : A^2 \mapsto \{M, U\}$, um classificador booleano que utiliza uma combinação linear ou não linear λ de evidências de um conjunto $S_E = \{E_{<f_i,p_j>} \mid f_i \in F_s, p_j \in P(A)\}$ e um ponto de corte $c \in \mathbb{R}$.

A Equação (2.3) mostra a combinação de evidências utilizadas na regra de ligação da Equação (2.4) para efetuar a deduplicação da base de dados da Tabela 2 e suas respectivas evidências na Tabela 4.

$$\lambda(r_a, r_b) = E_{<n,JW>}(r_a, r_b) + E_{<sn,JW>}(r_a, r_b) \tag{2.3}$$

Tabela 4 – Conjunto de valores de evidências da base de dados da Tabela 2 utilizando as funções Levenshtein (L) e Jaro-Winkler (JW). Os nomes dos atributos foram contraídos (n para *nome*, sn para *sobrenome* e num para *numero*).

r_i	r_j	$E_{<n,L>}$	$E_{<sn,L>}$	$E_{<num,L>}$	$E_{<n,JW>}$	$E_{<sn,JW>}$	$E_{<num,JW>}$
1	2	0.000	7.000	3.000	1.000	0.000	0.000
1	3	5.000	7.000	2.000	0.000	0.423	0.000
1	4	6.000	7.000	2.000	0.000	0.000	0.000
1	5	0.000	7.000	0.000	1.000	0.000	1.000
1	6	6.000	0.000	2.000	0.000	1.000	0.000
2	3	5.000	8.000	3.000	0.000	0.431	0.000
2	4	6.000	7.000	3.000	0.000	0.437	0.000
2	5	0.000	0.000	3.000	1.000	1.000	0.000
2	6	6.000	7.000	3.000	0.000	0.000	0.000
3	4	2.000	2.000	1.000	0.893	0.908	0.667
3	5	5.000	8.000	2.000	0.000	0.431	0.000
3	6	2.000	7.000	2.000	0.893	0.423	0.000
4	5	6.000	7.000	2.000	0.000	0.437	0.000
4	6	0.000	7.000	2.000	1.000	0.000	0.000
5	6	6.000	7.000	2.000	0.000	0.000	0.000

$$L_k(r_a, r_b) : \lambda(r_a, r_b) > 1.8 \tag{2.4}$$

Sendo assim, para classificar $(r_a, r_b) \in A^2$ no exemplo proposto, pode-se usar a regra de ligação L_k conforme as Equações (2.5) e (2.6). O componente λ da regra de ligação é obtido através de programação genética, utilizada para construir combinações de evidências com o intuito de melhorar o poder de classificação das regras de ligação. Em (CARVALHO et al., 2012) o ponto de corte $c \in \mathbb{R}$ é um valor fixo (3.0, por exemplo), porém, neste trabalho, o ponto de corte é estimado para cada regra de ligação (mais detalhes no Capítulo 3).

$$M = \{(r_a, r_b) \mid (r_a, r_b) \in A^2, a \neq b, L_k(r_a, r_b) = 1\} \tag{2.5}$$

$$U = \{(r_a, r_b) \mid (r_a, r_b) \in A^2, a \neq b, L_k(r_a, r_b) = 0\} \tag{2.6}$$

É possível notar que para as evidências apresentadas na Tabela 4, a regra de ligação ilustrada na Equação (2.4) classifica corretamente todos os pares de registros. O capítulo 3 detalha como a combinação de evidências da regra de ligação é encontrada. A Tabela 5 mostra a aplicação da regra de ligação da Equação (2.4) nos valores de evidências mostrado na Tabela 4 e a respectiva classificação de cada par de registros segundo a regra de ligação mencionada.

Tabela 5 – Evidências, suas respectivas combinações que compõem a regra de ligação na Equação (2.4) e sua classificação.

r_i	r_j	$E_{<n,JW>}$	$E_{<sn,JW>}$	$\lambda(r_a, r_b)$ (Eq. 2.3)	L_k (Eq. 2.4)	M ou U
1	2	1.000	0.000	1.000	0	U
1	3	0.000	0.423	0.423	0	U
1	4	0.000	0.000	0.000	0	U
1	5	1.000	0.000	1.000	0	U
1	6	0.000	1.000	1.000	0	U
2	3	0.000	0.431	0.431	0	U
2	4	0.000	0.437	0.437	0	U
2	5	1.000	1.000	**2.000**	**1**	M
2	6	0.000	0.000	0.000	0	U
3	4	0.893	0.908	**1.801**	**1**	M
3	5	0.000	0.431	0.431	0	U
3	6	0.893	0.423	1.316	0	U
4	5	0.000	0.437	0.437	0	U
4	6	1.000	0.000	1.000	0	U
5	6	0.000	0.000	0.000	0	U

As evidências apresentadas como resultados deste trabalho são geradas via programação genética, cujo conceito é apresentado na seção seguinte.

2.2 Programação genética

Os Algoritmos Genéticos (*Genetic Algorithms* - GA), desenvolvido inicialmente em (HOLLAND, 1975), são abordagens evolucionárias baseadas na teoria de evolução de Darwin, em que a solução de um problema por um algoritmo genético é encontrada através da evolução dos *indivíduos* de uma *população*. Cada indivíduo, que representa uma das possíveis soluções de um determinado problema, são compostos de células e cada célula possui o mesmo conjunto de um ou mais *cromossomos* (cadeias de DNA). Um cromossomo pode ser dividido em *genes*, que representam *características*, como "cor dos olhos"por exemplo. A *configuração* de uma característica(como por exemplo "cor dos olhos", pode ser "azul", "verde"ou "castanho") é chamada de *alelo*. Cada gene é localizado numa posição específica chamada *locus*. O conjunto completo de todos os cromossomos forma o *genoma* e o *genótipo* refere-se a um conjunto específico de genes dentro do genoma. Uma das representações computacionais mais simples de um indivíduo é dada por uma *string* de *bits*, como em "010011100".

Nos algoritmos genéticos, a população (conjunto de soluções candidatas) é evoluída através de um processo iterativo em que vários *operadores genéticos* são aplicados com o intuito de percorrer o espaço de busca das soluções. Os indivíduos são avaliados em relação ao seu valor de *fitness*, que denota o grau de satisfação do problema (ou, em

termos biológicos, a sua capacidade de se adaptar ao meio). Os operadores genéticos mais básicos segundo (MELANIE, 1999) são:

- ***Seleção*** - seleciona indivíduos para reprodução de acordo com o *fitness* de cada um;
- ***Cruzamento*** - seleciona aleatoriamente um locus entre dois indivíduos e combina o material genético desses dois indivíduos através da troca dos subconjuntos de genes antes e depois do locus, gerando assim dois novos indivíduos "filhos"com o material genético dos indivíduos anteriores;
- ***Mutação*** Escolhe um locus aleatório e faz uma troca do gene daquele locus. Na representação de *string* binária, a operação de mutação troca "0"por "1"e "1"por "0".

Segundo (MELANIE, 1999), um GA simples funciona da seguinte forma:

1. Inicie uma população de n indivíduos;
2. Calcule o *fitness* $f(x)$ de cada indivíduo na população;
3. Repita os passos seguintes até que n novos filhos sejam gerados dentro de uma nova população:
   a) **Selecione** um par de pais com chances proporcionais aos seus respectivos *fitness*;
   b) Faça o **cruzamento** desses dois pais para gerar dois filhos;
   c) Faça a **mutação** de cada locus de cada um dos filhos com probabilidade p_m (taxa de mutação);
   d) Adicione os novos filhos dentro de uma nova população.
4. substitua a população atual pela nova população;
5. Se o critério de parada não for atingido, volte ao passo 2. Caso contrário, apresente o melhor indivíduo da população.

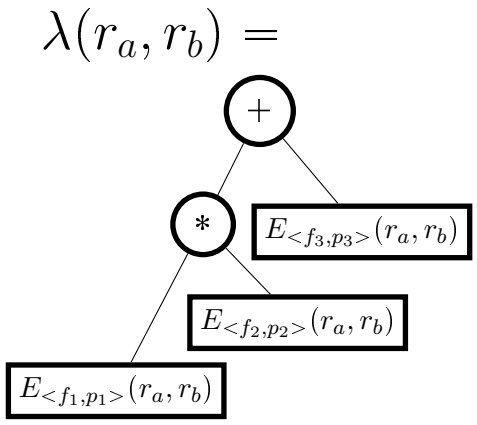
Os algoritmos genéticos são a base para uma versão mais específica de algoritmo evolucionário: a Programação Genética.

Programação Genética (*Genetic programming* - GP), segundo (KOZA, 1994), é uma técnica baseada em algoritmos genéticos para resolução de problemas através da evolução de programas de computador por meio de seleção natural que produzem saídas desejadas a partir de entradas específicas.

A programação genética tem sido aplicada em diferentes tipos de problemas, como otimização de redes de comunicação via celular (FENTON et al., 2017), cálculos de estratégias de seguradoras (DEHGHANPOUR; ESFAHANIPOUR, 2017), segurança de redes (MRUGALA; TUPTUK; HAILES, 2017) e principalmente como classificação (ESPEJO; VENTURA; HERRERA, 2010).

Dentre as diferenças entre GP e GA, encontra-se a representação do indivíduo, em que na GP o indivíduo é representado por uma estrutura de dados em formato de árvore, que compõe uma hierarquia de funções primitivas e terminais apropriados para o domínio específico do problema. A figura 2 ilustra um exemplo de uma função graficamente ilustrado como uma árvore. Dessa forma, é possível evoluir classificadores eficientes utilizando GP (CARVALHO et al., 2012).

Figura 2 – Representação gráfica de um indivíduo λ em forma de árvore



No processo evolutivo da GP, os indivíduos são manipulados por operadores genéticos, tais como os operadores de cruzamento, seleção e mutação (KOZA, 1994). Esses operadores são responsáveis por tentar, de forma iterativa, encontrar melhores indivíduos (que são soluções para o problema em questão) e repassar esses indivíduos para as gerações subsequentes.

Na GP, a geração de indivíduos pode ser aleatória, pode seguir alguma heurística ou pode ser híbrida. Os indivíduos da GP tem a desvantagem da complexidade devido à estrutura de dados que utilizam (árvores) e à combinação de nós, que torna necessário a apresentação de um exemplo de geração de um indivíduo. O Algoritmo 1 mostra um exemplo de geração de indivíduo aleatório.

O parâmetro O_p é um conjunto de operadores primitivos, tais como $\{+, -, /, *, sin, cos, exp, log$ enquanto que T é um conjunto de nós terminais (que neste caso podem ser apenas valores absolutos, como $\{1, 2, 4.5, 10\}$. A *aridade* comentada no código, diz respeito à quantidade de parâmetros que as funções em O_p irão receber: os passos entre a linha 11 e 15 garantem que a aridade das funções sejam respeitadas e que por isso não haja erros na sua invocação.

Algorithm 1 Exemplo de geração de indivíduo aleatório

```
1: function IndividuoAleatorio(O_p, T)
2:     if O_p = ∅ then
3:         return EscolhaAleatoria(T)
4:     end if
5:     if chance ⩾ 50% then
6:         return EscolhaAleatoria(T)
7:     else
8:         no ← EscolhaAleatoria(O_p)
9:         O'_p ← O_p\no
10:        aridade ← 0              ▷ Aridade é o número de parâmetros de uma função
11:        while aridade < no.aridade do
12:            no_filho ← IndividuoAleatorio(O'_p, T)
13:            InsereNo(no, no_filho)
14:            aridade ← aridade + 1
15:        end while
16:        return no
17:    end if
18: end function
```

A seleção de pais para cruzamento (passo 4) é um operador que visa escolher os indivíduos que serão recombinados no cruzamento. Dentre as estratégias de seleção mais comuns, encontram-se a seleção por roleta e torneio de indivíduos.

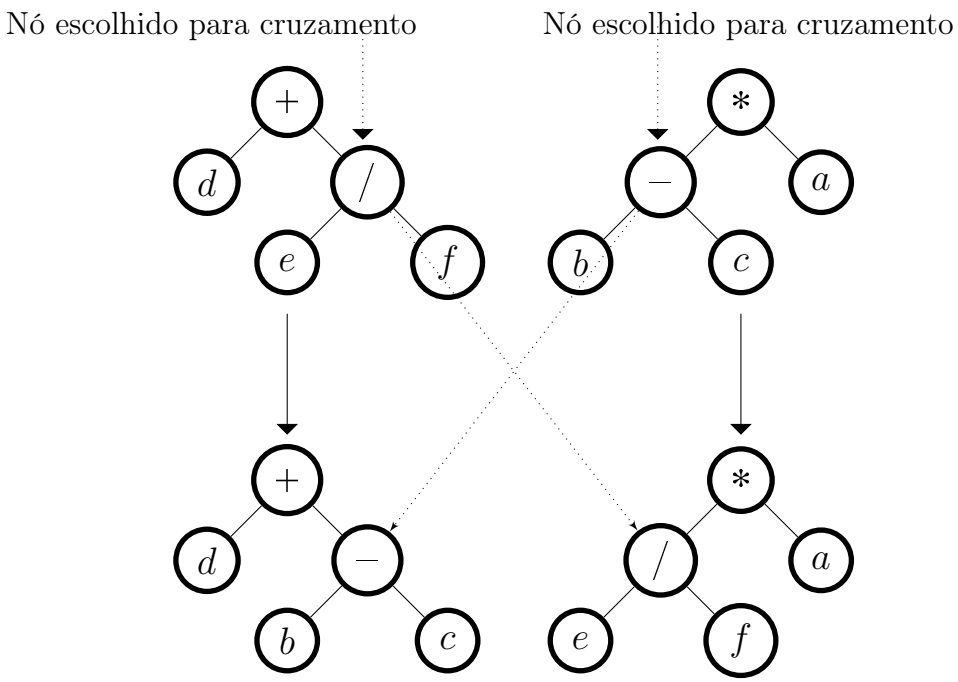
O operador de cruzamento (passo 5) cria novos programas (indivíduos) através de combinações de partes de indivíduos já existentes. Essas partes são, na verdade, subárvores de dois indivíduos que, quando combinadas, formam um novo indivíduo. Pelo fato de que subárvores inteiras são trocadas entre os indivíduos, o operador de cruzamento sempre produz indivíduos válidos independentemente da escolha dos pontos de cruzamento (KOZA, 1994). A Figura 3 ilustra graficamente o operador de cruzamento.

O operador de mutação (passo 6) é responsável por gerar novas soluções no espaço de busca através da modificação de um gene aleatório do indivíduo, em que um operador ou um nó é alterado, evitando uma convergência prematura. A chance de um indivíduo sofrer uma mutação é determinada pela taxa de mutação escolhida na GP e o nó a ser alterado é escolhido aleatoriamente dentre o conjunto de nós do indivíduo. Respeitando a correta estrutura do indivíduo, o operador de mutação, tal qual o operador de cruzamento, também gera programas válidos. A figura 4 ilustra graficamente o operador de mutação.

A estratégia de substituição da população (passo 7) pode ser elitista ou não elitista. A estratégia elitista, mais comumente usada, garante que o melhores indivíduos sigam inalterados para a próxima geração.

Apesar de (CARVALHO et al., 2012) mostrar classificadores construídos apenas com a abordagem baseada em GP, ainda há formas de melhorar os resultados conseguidos

Figura 3 – Representação gráfica do operador de cruzamento.



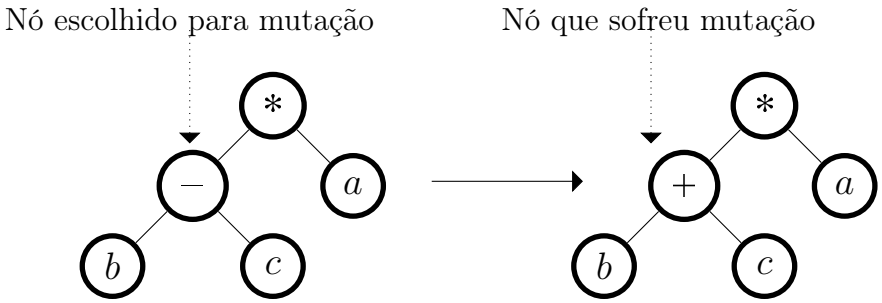


Figura 4 – Representação gráfica do operador de mutação

com essa abordagem. Técnicas como construção indutiva evolucionária (*Evolutionary Constructive Induction* - ECI), descrita em (MUHARRAM; SMITH, 2005), combinam os resultados obtidos com a abordagem da GP com outros classificadores, como árvores de decisão[4]. As próximas seções mostram como as árvores de decisão e a construção indutiva evolucionária funcionam.

2.3 Indução de árvores de decisão

Uma árvore de decisão é uma forma de expressar uma regra de classificação (QUINLAN, 1986). A indução de uma árvore de decisão, por sua vez, é o processo de aprendizado que desenvolve uma árvore de decisão a partir de exemplos em um processo guloso[5], *top-*

[4] O algoritmo ID3, descrito em (QUINLAN, 1986), é um exemplo.

[5] Um algoritmo guloso é um paradigma de solução de problemas que se baseia na escolha da solução ótima local em cada iteração da busca pela solução final.

down, sem fazer correções durante o treinamento. Tal característica é um contraste com outros algoritmos utilizados para classificação, como é o caso das redes neurais artificiais.

Por ser um algoritmo guloso, o processo de indução de árvores de decisão precisa encontrar soluções ótimas locais na construção de cada nó da árvore. Para encontrar tais soluções ótimas locais, o critério de *ganho de informação* é utilizado para o particionamento da base de dados. Para melhor compreensão do conceito de ganho de informação, é necessário detalhar um conceito dependente, a *entropia* da base de dados.

A entropia de uma base de dados é caracterizada, informalmente, pelo nível de balanceamento de classes de uma base de dados, utilizando um valor no intervalo real $[0, 1]$: quanto mais próximo de 1, mais balanceado. Em uma base de dados com classificação binária, por exemplo, a entropia 1 se houvesse 50% de elementos da classe A e 50% de elementos da classe B. De maneira formal, a entropia de uma base de dados S é definida conforme a Equação (2.7).

$$E(S) = \sum_{i}^{c} -p_i log_2 p_i \tag{2.7}$$

Em que c é o conjunto de classes da base de dados S e p_i é a proporção de exemplos da classe i (MITCHELL, 1997).

O ganho de informação é uma medida estatística usada para determinar qual o melhor atributo classifica subconjunto da base de treinamento. Em outras palavras, o ganho de informação de um determinado atributo mede o potencial de redução da *entropia* da base de dados caso esse atributo A seja utilizado para particionar a base de dados. A Equação (2.8) ilustra o cálculo do ganho de informação do atributo A dado a base de dados S.

$$G(S, a) = E(S) - \sum_{v \in Values(a)} \frac{|S_{av}|}{|S|} E(S_{av}) \tag{2.8}$$

Em que $Values(a)$ é o conjunto de todos valores possíveis do atributo a e S_{av} é um subconjunto de S em que o atributo a tem valor v, ou seja:

$$S_{av} = \{s \in S \mid A(s) = v\} \tag{2.9}$$

Para melhor ilustração do processo de indução de árvores de decisão, o algoritmo ID3 é apresentado no Algoritmo (2).

Os detalhes do algoritmo são expostos a seguir:

1. A função $CondicaoParada$ (linha 2) é utilizada para determinar se é possível que a

Algorithm 2 Pseudocódigo do ID3 (adaptado de (TAN, 2006))

```
1: function ID3(S, A)
2:     if CondicaoParada(S,A) then
3:         folha ← CriaNo()
4:         folha.rotulo ← Classifica(S, A)
5:         return folha
6:     else
7:         raiz ← CriaNo()
8:         raiz.attr ← arg max_a G(S, a), ∀a ∈ A
9:         for v ∈ Values(raiz.attr, S) do
10:            S_av = {s | s ∈ S and !ContemValor(raiz.attr, s, v)}
11:            no_filho = ID3(S_av, A)
12:            Adicione no_filho em raiz
13:        end for
14:        return raiz
15:    end if
16: end function
```

árvore "aprenda" com os exemplos S restantes. A condição de parada mais simples é ter poucos exemplos de treinamento ou todos eles pertencerem a uma mesma classe;

2. A função $CriaNo$ (linha 3) é responsável por instanciar/alocar a estrutura de dados dos nós da árvore. Um nó na árvore possui um rótulo de classe *rotulo* ou um nome de atributo *attr*;

3. A função $Classifica$ (linha 4) é utilizada para determinar o rótulo da folha da árvore, dado o conjunto de dados faltantes para serem classificados. Essa função basicamente toma o rótulo com o valor mais comum em S;

4. A linha 8 escolhe, dentre todos os atributos $a \in A$, o que tem maior ganho de informação (veja a Equação 2.8);

5. A função $Values$ (linha 9) retorna os possíveis valores para o atributo $raiz.attr$ dentro da base de dados S;

6. Na linha 10, $S_{av} \subset S$ é um subconjunto de S que não contém o valor v para o atributo $raiz.attr$. Isso evidencia a natureza gulosa do algoritmo;

2.3.1 Um exemplo prático de indução de árvores de decisão

Para ilustrar a abordagem de indução de árvores de decisão apresentada na Seção 2.3, mostra-se o exemplo seguinte adaptado de (MITCHELL, 1997) para o problema de deduplicação de registros. Para efeitos de brevidade e simplificação, os atributos foram discretizados e dicotomizados (assumem apenas dois valores) através do uso de regras de ligação da seguinte forma:

- $L_a(r_i, r_j) : E_{<n,L>}(r_i, r_j) \leqslant 2$
- $L_b(r_i, r_j) : E_{<sn,L>}(r_i, r_j) \leqslant 2$
- $L_c(r_i, r_j) : E_{<num,L>}(r_i, r_j) \leqslant 3$
- $L_d(r_i, r_j) : E_{<n,JW>}(r_i, r_j) \geqslant 0.893$
- $L_e(r_i, r_j) : E_{<sn,JW>}(r_i, r_j) \geqslant 0.908$
- $L_f(r_i, r_j) : E_{<sn,JW>}(r_i, r_j) \geqslant 0$

Portanto, a base de dados de treinamento de acordo com a transformação citada fica representada de acordo com a Tabela 6. Essa nova base de dados será utilizada no restante deste exemplo de ilustração de construção de uma árvore de decisão.

Tabela 6 – Base de dados baseada na tabela 4 e nas regras de ligação apresentadas e sua respectiva classificação.

(r_i, r_j)	$L_a(r_i, r_j)$	$L_b(r_i, r_j)$	$L_c(r_i, r_j)$	$L_d(r_i, r_j)$	$L_e(r_i, r_j)$	$L_f(r_i, r_j)$	M ou U
(1,2)	1	0	1	1	0	1	U
(1,3)	0	0	1	0	0	1	U
(1,4)	0	0	1	0	0	1	U
(1,5)	1	0	1	1	0	1	U
(1,6)	0	1	1	0	1	1	U
(2,3)	0	0	1	0	0	1	U
(2,4)	0	0	1	0	0	1	U
(2,5)	1	1	1	1	1	1	M
(2,6)	0	0	1	0	0	1	U
(3,4)	1	1	1	1	1	1	M
(3,5)	0	0	1	0	0	1	U
(3,6)	1	0	1	1	0	1	U
(4,5)	0	0	1	0	0	1	U
(4,6)	1	0	1	1	0	1	U
(5,6)	0	0	1	0	0	1	U

O primeiro passo do algoritmo ID3 descrito no Algoritmo 2 é verificar a condição de parada. Como a base de dados da Tabela 6 possui mais de uma classe (M e U), então a entropia é maior que 0, o critério de parada não foi atingido e o algoritmo pode continuar na escolha do melhor atributo para classificação.

A escolha do atributo que melhor classifica a base de dados atual é feito através do cálculo de ganho de informação de todos os atributos. Segundo a Equação (2.8), é necessário calcular a entropia da base de dados e depois subtrair essa entropia pela soma das entropias de um determinado atributo. A base de dados na Tabela 6 possui 15 exemplos de treinamento, sendo que 2 deles são positivos e 13 negativos. Portanto:

$$E(S) = \sum_{i}^{\{M,U\}} -p_i log_2 p_i \tag{2.10}$$

$$E(S) = (-p_M log_2 p_M) + (-p_U log_2 p_U) \tag{2.11}$$

$$E(S) = (-(2/15) log_2 (2/15)) + (-(13/15) log_2 (13/15)) \tag{2.12}$$

$$E(S) = 0.387 + 0.178 \tag{2.13}$$

$$E(S) = 0.566 \tag{2.14}$$

Com o valor da entropia da base de dados, é possível prosseguir com o cálculo do ganho de informação dos atributos.

$$G(S, L_a) = E(S) - \sum_{v \in Values(L_a)} \frac{|S_{L_a v}|}{|S|} E(S_{L_a v}) \tag{2.15}$$

$$G(S, L_a) = E(S) - \sum_{v \in 0,1} \frac{|S_{L_a v}|}{|S|} E(S_{L_a v}) \tag{2.16}$$

$$G(S, L_a) = E(S) - \left\{ \left(\frac{|S_{L_a 0}|}{|S|} E(S_{L_a 0}) \right) + \left(\frac{|S_{L_a 1}|}{|S|} E(S_{L_a 1}) \right) \right\} \tag{2.17}$$

$$G(S, L_a) = 0.566 - \left\{ \left(\frac{|9|}{|15|} 0.000 \right) + \left(\frac{|6|}{|15|} 0.918 \right) \right\} \tag{2.18}$$

$$G(S, L_a) = 0.198 \tag{2.19}$$

O mesmo procedimento aplicado entre as Equações (2.15) e (2.19) deve ser aplicado a todos os atributos a fim de descobrir o que tem o maior ganho de informação. O resultado obtido é o seguinte:

- $G(S, L_a) = 0.198$
- $G(S, L_b) = 0.382$
- $G(S, L_c) = 0.000$
- $G(S, L_d) = 0.198$
- $G(S, L_e) = 0.382$
- $G(S, L_f) = 0.000$

É possível perceber que qualquer um dos atributos L_b ou L_e podem ser usados como a raiz da árvore de decisão. Para este exemplo, toma-se L_b como raiz. A partir daí, para cada possível valor de L_b (ou seja, 0,1), uma nova ramificação da árvore será criada. Para o $L_b = 0$ temos S_{L_b0} conforme representado na Tabela 7 e para $L_b = 1$ temos S_{L_b1} ilustrado na Tabela 8.

Tabela 7 – Base de dados filtrada para todo $L_b = 0$.

(r_i, r_j)	$L_a(r_i, r_j)$	$L_b(r_i, r_j)$	$L_c(r_i, r_j)$	$L_d(r_i, r_j)$	$L_e(r_i, r_j)$	$L_f(r_i, r_j)$	M ou U
(1,2)	1	0	1	1	0	1	U
(1,3)	0	0	1	0	0	1	U
(1,4)	0	0	1	0	0	1	U
(1,5)	1	0	1	1	0	1	U
(2,3)	0	0	1	0	0	1	U
(2,4)	0	0	1	0	0	1	U
(2,6)	0	0	1	0	0	1	U
(3,5)	0	0	1	0	0	1	U
(3,6)	1	0	1	1	0	1	U
(4,5)	0	0	1	0	0	1	U
(4,6)	1	0	1	1	0	1	U
(5,6)	0	0	1	0	0	1	U

Tabela 8 – Base de dados filtrada para todo $L_b = 1$

(r_i, r_j)	$L_a(r_i, r_j)$	$L_b(r_i, r_j)$	$L_c(r_i, r_j)$	$L_d(r_i, r_j)$	$L_e(r_i, r_j)$	$L_f(r_i, r_j)$	M ou U
(1,6)	0	1	1	0	1	1	U
(2,5)	1	1	1	1	1	1	M
(3,4)	1	1	1	1	1	1	M

Nota-se que $E(S_{L_b0}) = 0$; portanto, o critério de parada é atingido quando $L_b = 0$ e o próximo nó conterá o rótulo U, que é a única classe restante. Por outro lado, para $L_a = 1$ a entropia $E(S_{L_a1}) = 0.918$, o que indica que o critério de parada ainda não foi atingido e o algoritmo pode continuar na escolha do atributo com mais ganho de informação. A situação atual do ganho de informação dos atributos dado a base de dados filtrada descrita na Tabela **??** é:

- $G(S, L_a) = 0.918$
- $G(S, L_b) = 0.000$
- $G(S, L_c) = 0.000$
- $G(S, L_d) = 0.000$
- $G(S, L_e) = 0.918$
- $G(S, L_f) = 0.000$

Novamente, existem dois atributos que possuem os valores mais altos iguais. Escolhendo L_a e continuando com o processo até a condição de parada final, têm-se a árvore de decisão ilustrada na Figura 5.

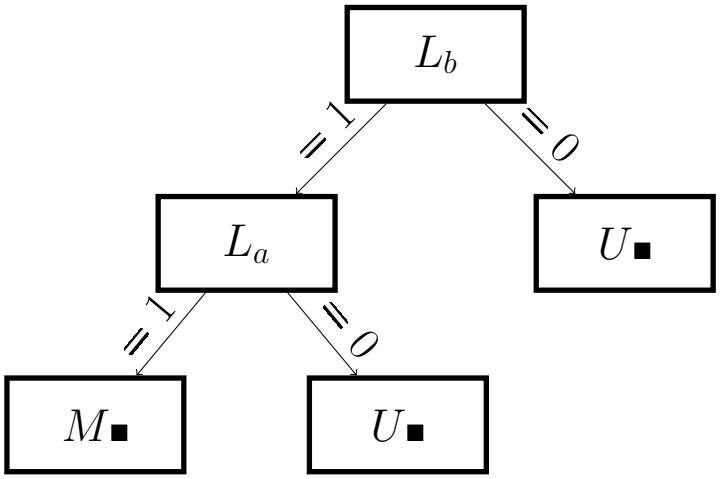


Figura 5 – Ilustração da árvore de decisão final gerada pelo Algoritmo 2 para classificação da base de dados ilustrada na Tabela 6

2.3.2 Discretização de atributos contínuos

As árvores de decisão até agora apresentadas funcionam para atributos discretos. No entanto, depara-se com o problema de classificar atributos reais. Para que seja possível a classificação de tais atributos por árvores de decisão, é necessário discretizar esses valores. Dentre as técnicas comuns empregadas para esse fim, encontra-se a discretização de intervalos (utilizando pontos de corte) e uma *condição lógica* (ex.: $>, \geqslant, <, \leqslant, \neq$) (FAYYAD; IRANI, 1993)

Em (FAYYAD; IRANI, 1993) é provado que um ponto de corte que minimiza a entropia E **sempre** será uma fronteira e que a classificação terá a maior precisão possível.

Para que seja possível o uso de árvores de decisão neste trabalho, uma técnica conhecida como Construção Indutiva Evolucionária precisa ser aplicada. A próxima seção provê mais detalhes sobre essa técnica.

2.4 Construção indutiva evolucionária

A Construção Indutiva Evolucionária (*Evolutionary Constructive Induction* - ECI) é consiste no processo de criação de novos atributos construídos a partir da base de dados de treinamento que precede o processo de classificação por árvores de decisão e foi descrito inicialmente em (MUHARRAM; SMITH, 2005). O objetivo da ECI é aumentar a variedade de atributos e, como resultado, melhorar os resultados da classificação.

Segundo (MUHARRAM; SMITH, 2005), se o conjunto dos atributos disponíveis não incluir um ou mais atributos poderosamente previsíveis, o poder de classificação de qualquer algoritmo que não combine esses atributos será prejudicado. Portanto, o ECI pode ser um passo importante na fase de pré-processamento dos dados em bases que não apresentam um ou mais atributos poderosos em termos de previsibilidade.

Os novos atributos são gerados utilizando programação genética, que produz combinações lineares e não lineares, sendo que os resultados dessa abordagem descritos em (MUHARRAM; SMITH, 2005), mostram que todos os algoritmos de classificação utilizados tiveram resultados melhores com o uso de ECI, mesmo quando um único novo atributo foi adicionado.

A Tabela 9 ilustra a aplicação de ECI em uma base de dados descrita anteriormente.

r_i	r_j	(1)	(2)	(3)	(4)	(5)	(6)	(4)+(5)
1	2	0.000	7.000	3.000	1.000	0.000	0.000	1.000
1	3	5.000	7.000	2.000	0.000	0.423	0.000	0.423
1	4	6.000	7.000	2.000	0.000	0.000	0.000	0.000
1	5	0.000	7.000	0.000	1.000	0.000	1.000	1.000
1	6	6.000	0.000	2.000	0.000	1.000	0.000	1.000
2	3	5.000	8.000	3.000	0.000	0.431	0.000	0.431
2	4	6.000	7.000	3.000	0.000	0.437	0.000	0.437
2	5	0.000	0.000	3.000	1.000	1.000	0.000	2.000
2	6	6.000	7.000	3.000	0.000	0.000	0.000	0.000
3	4	2.000	2.000	1.000	0.893	0.908	0.667	1.575
3	5	5.000	8.000	2.000	0.000	0.431	0.000	0.431
3	6	2.000	7.000	2.000	0.893	0.423	0.000	1.316
4	5	6.000	7.000	2.000	0.000	0.437	0.000	0.437
4	6	0.000	7.000	2.000	1.000	0.000	0.000	1.000
5	6	6.000	7.000	2.000	0.000	0.000	0.000	0.000

Tabela 9 – Base de dados da Tabela 4 complementada com uma nova propriedade (4)+(5) gerada via programação genética. Os nomes das propriedades foram contraídos por questões de espaço.

A seguir, a relação de nomes das propriedades abreviadas na Tabela 9:

- (1) - $E_{n,L}$;
- (2) - $E_{sn,L}$;
- (3) - $E_{num,L}$;
- (4) - $E_{n,JW}$;
- (5) - $E_{sn,JW}$;

- (6) - $E_{num,JW}$;
- (4)+(5) - $E_{n,JW} + E_{sn,JW}$;

Percebe-se que o poder de classificação do atributo gerado $\lambda = E_{n,JW} + E_{sn,JW}$ é ótimo caso utilizado numa regra de ligação $L(r_i, r_j) : \lambda(r_i, r_j) \geqslant 1.575$. O mesmo poder de classificação ótimo do exemplo só foi alcançado com o uso de indução de árvores de decisão, descrita na seção 2.3.1.

Espera-se que os conceitos apresentados neste capítulo dêem uma fundamentação suficiente para o bom entendimento do próximo capítulo, em que são apresentadas as metodologias utilizadas neste trabalho.

3 Abordagem utilizada

Este capítulo apresenta em detalhes a abordagem desenvolvida para deduplicação de registros. As diferenças deste trabalho para (CARVALHO et al., 2012) resumem-se a:

- **Nova heurística de geração de população inicial:** esta nova heurística toma como princípio norteador o fato que o potencial de desbalanceamento dos atributos das bases de dados é alta, ou seja, que há regras de ligação baseadas em um único atributo com grande poder de classificação;
- **Cálculo do ponto de corte:** o ponto de corte é fixo em outras regras de ligação descritas em (CARVALHO et al., 2012). Este trabalho utiliza uma abordagem de cálculo de ponto de corte que sempre encontra o melhor ponto de corte para o indivíduo, o que dispensa o uso de folhas com valores aleatoriamente escolhidos;
- **Uso de Construção Indutiva Evolucionária:** permite que os indivíduos sejam transformados em novas propriedades da base de dados e permite o uso de outros classificadores para potencializar ainda mais o poder de ligação já obtido.

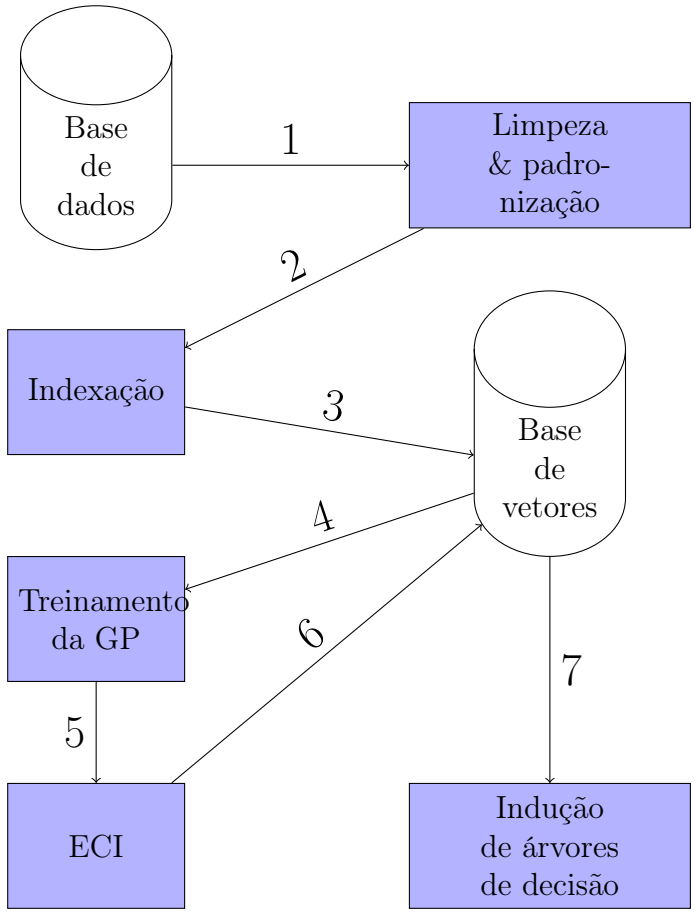
A Figura 6 ilustra graficamente o funcionamento da abordagem utilizada. Com essa representação gráfica, é possível notar que a base de dados original é utilizada apenas para a produção da base de dados de vetores de classificação, e que esta base de dados é utilizada para o processo de treinamento dos classificadores. É importante observar que dados são inseridos na base de dados de vetores mais de uma vez: depois que a indexação é feita (passo 3 na figura) e depois que novos atributos são construídos via ECI e então armazenados nessa base de dados (passo 6).

O **Algoritmo 3** é um pseudocódigo possui quatro parâmetros, sendo A a base de dados, F_s o conjunto de funções de similaridade entre strings, N_{pop} o tamanho da população e M_{gen} a quantidade de gerações.

Na linha 2, o algoritmo parte de uma população inicial gerada através do mecanismo especificado no Algoritmo 4. Cada indivíduo $p \in pop$, é um programa λ, que posteriormente será utilizado para formar regras de ligação. A Seção 3.1 detalha a heurística de geração da população inicial.

Na linha 3, cada individuo da população inicial é avaliado, tomando como base o conjunto de dados A, contendo os pontos de corte computados dentro da função $CalculaFitness$. A avaliação por fitness constrói as regras de ligação a partir dos indivíduos e aplica essas regras de ligação na base de dados de treinamento. A Seção 3.2 mostra como o ponto de corte e o fitness são calculados.

Figura 6 – Representação gráfica do processo de deduplicação de registros desenvolvido neste trabalho



Algorithm 3 Algoritmo de deduplicação proposto

1: **function** ADGPDedup(A, F_s, N_{pop}, M_{gen})
2: $pop \leftarrow GeraPopulacaoInicial(A, F_s, N_{pop})$
3: $pop \leftarrow CalculaFitness(pop, A)$
4: $gen \leftarrow 0$
5: **while** $MaxF_1(pop) \leqslant 1.0$ **and** $gen \leqslant M_{gen}$ **do**
6: $pop \leftarrow GenSolucoesGP(pop, F_s, N_pop)$
7: $gen \leftarrow gen + 1$
8: **end while**
9: $A_2 \leftarrow ComplementaDataset(A^2, pop)$
10: $dec_{tree} \leftarrow ArvoreDecisao(A_2)$
11: **return** dec_{tree}
12: **end function**

Entre as linhas 4 e 8, acontece o processo de evolução dos indivíduos da GP, onde são aplicados os operadores de seleção, cruzamento e mutação, utilizando o critério de parada de quantidade máxima de gerações ou *fitness* perfeito.

A linha 9 utiliza a população gerada para complementar a base de dados com novos atributos através do uso do ECI[1], que transforma os indivíduos gerados em atributos

[1] Veja a Seção 2.4 para maiores detalhes

extras com maior poder de classificação para a base de dados de treinamento da árvore de decisão.

A linha 10 efetua o aprendizado da árvore de decisão utilizando o algoritmo CART[2] e finalmente a linha 11 finaliza a execução do algoritmo devolvendo a árvore de decisão aprendida.

Os pontos chave do algoritmo serão demonstrados nas seções seguintes.

3.1 Heurística de geração de população inicial

Uma característica importante da abordagem de cálculo de ponto de corte utilizada neste trabalho é que ela potencializa o poder de classificação de uma regra de ligação com apenas uma evidência. Sendo assim, uma abordagem óbvia para geração da população inicial é a geração de indivíduos com apenas uma evidência. O **Algoritmo 4** ilustra o funcionamento da heurística de geração de população inicial.

Algorithm 4 Heurística de geração de população inicial

```
1: function GeraPopulacaoInicial(A, F_s, N_pop)
2:     pop ← ∅
3:     for all f ∈ F_s do
4:         for all p ∈ P(A) do
5:             pop ← pop ∪ {E_<f,p>}
6:         end for
7:     end for
8:     if |pop| < N_pop then
9:         for all i ∈ {|pop| ... N_pop} do
10:            pop ← pop ∪ {IndividuoAleatorio()}
11:        end for
12:    end if
13:    return pop
14: end function
```

No algoritmo, *IndividuoAleatorio* (linha 10) é uma função para gerar um indivíduo com apenas um operador e um ou dois terminais escolhidos aleatoriamente[3]. Em GPs tradicionais, N_{pop} indivíduos seriam gerados utilizando o método de geração de indivíduo aleatório, mas a abordagem descrita neste trabalho apenas utiliza esse recurso em casos excepcionais.

A heurística de geração da população inicial não cria sempre N_{pop} indivíduos, mas sim $max(N_{pop}, |P_a| * |F_s|)$. A intenção desta abordagem é explorar ao máximo o espaço de busca de indivíduos com apenas uma evidência. A quantidade máxima de indivíduos N_{pop} é aplicada no elitismo do algoritmo.

[2] Do inglês *Classification And Regression Tree*.
[3] Para maiores detalhes, consulte o Algoritmo 1 na Seção 2.2

3.2 Cálculo de *fitness*

A função de *fitness* utilizada é o escore F1, mesma métrica utilizada em (CARVALHO et al., 2012) e que é comumente utilizada para avaliar a qualidade dos classificadores. O escore F1 (Equação 3.1) é a média harmônica entre a precisão (Equação 3.2), relação entre os verdadeiros positivos T_p e os falsos positivos F_p – probabilidade de uma classificação positiva ser realmente positiva, e a revocação (Equação 3.3), relação entre os verdadeiros positivos e os falsos negativos F_n – probabilidade de se encontrar todos os itens realmente positivos.

$$F1 = 2 * \left(\frac{P * R}{P + R} \right) \tag{3.1}$$

$$P = \frac{T_p}{T_p + F_p} \tag{3.2}$$

$$R = \frac{T_p}{T_p + F_n} \tag{3.3}$$

Algorithm 5 Cálculo de fitness

```
1: function CalculaFitness(pop, A)
2:     pop' ← pop
3:     S_a ← blocagem(A)
4:     for all λ ∈ pop' do
5:         F1_max ← 0
6:         for all (r_a, r_b) ∈ S_a do                ▷ Ponto de corte
7:             v ← λ(r_a, r_b)
8:             L_kv ← Tree("λ(r_a, r_b) ⩾ v")
9:             if F_1(L_kv) > F1_max then
10:                F1_max ← F_1(L_kv)
11:            end if
12:        end for
13:        λ.f_1 = F1_max
14:    end for
15: return pop'
16: end function
```

No Algoritmo 5, S_a é um subconjunto de A^2 determinado por um método de blocagem contendo todos os exemplos rotulados positivos combinado com 5% de todos os exemplos negativos e L_{max} é uma regra de ligação cujo valor de fitness é máximo.

3.3 Ponto de corte

A abordagem descrita em (CARVALHO et al., 2012) utiliza os indivíduos gerados pela GP como combinadores lineares e não lineares das evidências encontradas entre

dois registros diferentes das bases de dados. Essa abordagem utiliza apenas operadores matemáticos como funções primitivas dos indivíduos e o valor das evidências e possíveis valores aleatórios nas extremidades (folhas) dos indivíduos. O indivíduo, então, representa uma função matemática cuja entrada são os valores das evidências de um par de registros e a saída é um valor $v \in \mathbb{R}$. O valor v é, então, comparado com um ponto de corte com valor fixo e determinado previamente para realizar a classificação e a GP ajusta-se adicionando ou removendo valores fixos das folhas dos indivíduos.

Neste trabalho, o ponto de corte é determinado para cada indivíduo, de forma dinâmica, como apresentado na linha 6 do Algoritmo 5. Esse modelo foi inspirado na forma como as árvores de decisão discretizam e criam pontos de corte para atributos contínuos ou com um grande intervalo de valores. De forma análoga às árvores de decisão, a classificação via regra de ligação necessita de um ponto de corte em que a classificação seja a melhor possível.

Como relatado na Seção 2.3.2, (FAYYAD; IRANI, 1993) provam que o ponto de corte encontrado pela maximização da entropia é o ponto de corte ideal para o atributo.

Outra característica importante da abordagem de ponto de corte proposto neste trabalho é que permite a simplificação do indivíduo, tornando desnecessário a inclusão de valores fixos para balancear o ponto de corte fixo. Por exemplo, considere a seguinte regra de ligação hipotética com um ponto de corte fixo em 0.95:

$$L_h(r_i, r_j) : E_{a_1,f_1}(r_i, r_j) + E_{a_2,f_2}(r_i, r_j) \geqslant 0.95 \tag{3.4}$$

Sabendo que o ponto de corte ideal está em 1.95, o lado esquerdo da inequação precisaria ser acrescido de 1.0 para ter a melhor classificação possível:

$$L_h(r_i, r_j) : E_{a_1,f_1}(r_i, r_j) + E_{a_2,f_2}(r_i, r_j) + 1.0 \geqslant 0.95 \tag{3.5}$$

O uso de operadores de mutação e cruzamento apropriados podem trazer o resultado desejado, porém, com um custo computacional mais variável e com uma maior incerteza.

3.4 Uso de construção indutiva evolucionária

A abordagem utilizada até agora é suficiente para produzir classificadores simples e, em sua maioria, lineares. Como forma de aumentar a capacidade de classificação da abordagem até aqui apresentada, este trabalho utiliza a capacidade de generalização do aprendizado de árvores de decisão (MITCHELL, 1997). Desta forma, as melhores combinações de evidências produzidas pela GP são inseridas como novas colunas na matriz

de vetores de similaridade,[4], que são posteriormente utilizadas no processo de aprendizado de árvores de decisão.

Todas as evidências geradas pela GP são utilizadas na árvore de decisão, deixando para esta o trabalho de escolher as propriedades mais adequadas para construção da árvore. A Seção 2.4 provê maiores detalhes neste aspecto.

3.5 Conclusão

Este capítulo detalhou a abordagem utilizada neste trabalho e evidenciou as diferenças metodológicas em relação ao trabalho de (CARVALHO et al., 2012), que se resumem ao uso de uma nova abordagem de cálculo do ponto de corte para as regras de ligação geradas pelo algoritmo de programação genética e uma nova abordagem para geração de população inicial. Acrescenta-se a isso, o uso de Construção Indutiva Evolucionária e de árvores de decisão com o intuito de potencializar o poder de classificação das regras de ligação geradas nos passos anteriores.

A avaliação prática da metodologia sugerida neste capítulo é discutida no próximo capítulo, que descreverá todos os experimentos realizados e seus respectivos resultados e discussões.

[4] O processo de classificação é aplicado sobre essa matriz. Para mais detalhes, consulte a Seção 2.1.

4 Experimentos e discussão

Nesta seção, nós apresentamos e discutimos os resultados dos experimentos realizados para avaliar nossa proposta de deduplicação de registros utilizando apenas classificadores construídos com GP e classificadores construídos com GP e árvores de decisão. Os experimentos realizados foram:

1. GP de apenas um atributo. O objetivo desse experimento é estabelecer um patamar de classificação com os classificadores mais simples possíveis de serem gerados;

2. GP de um ou mais atributos. O objetivo desse experimento é verificar a capacidade do GP de combinar um ou mais atributos face a classificação feita com apenas um atributo. Esse experimento também compara os resultados deste trabalho com outros trabalhos na literatura que utilizam abordagens baseadas em GP para deduplicação de registros;

3. Árvore de decisão com GP de um ou mais atributos. Esse experimento tem o objetivo de avaliar a principal contribuição deste trabalho analisando as melhorias que a geração de indivíduos com GP pode trazer para classificadores baseados em árvores de decisão e compara com os resultados obtidos por outros trabalhos da literatura.

Para avaliar as abordagens apresentadas neste trabalho, foram escolhidas bases de dados utilizadas em outros trabalhos de referência na literatura como em (CARVALHO et al., 2012). Utiliza-se três bases de dados sintéticas geradas através do sistema FEBRL [1] (CHRISTEN, 2008) e outras duas bases de dados reais[2]. As bases de dados sintéticas tem as seguintes configurações:

1. **Base de dados sintética 1:** Um único arquivo com 500 registros (400 originais, 100 duplicados), com um registro tendo até 5 duplicatas (escolhidos aleatoriamente seguindo a distribuição de Poisson) e com no máximo 2 modificações num único atributo e no máximo 2 por registro.

2. **Base de dados sintética 2:** Um único arquivo com 500 registros (350 originais, 150 duplicados), com um registro tendo até 5 duplicatas (escolhidos aleatoriamente

[1] Do Inglês: *Freely Extensible Biomedical Record Linkage*

[2] Não há referências em (CARVALHO et al., 2012) sobre a fonte dessas bases de dados. Este trabalho utiliza as bases de dados disponíveis em <http://www.cs.utexas.edu/users/ml/riddle/data.html> - Acesso em 30/11/2017

seguindo a distribuição de Poisson) e com no máximo 2 modificações num único atributo e no máximo 4 por registro.

3. **Base de dados sintética 3:** Um único arquivo com 500 registros (300 originais, 200 duplicados), com um registro tendo até 7 duplicatas (escolhidos aleatoriamente seguindo a distribuição de Poisson) e com no máximo 4 modificações num único atributo e no máximo 5 por registro.

As bases de dados sintéticas possuem, cada uma, 10 atributos. As duas bases de dados reais utilizadas possuem as seguintes características:

1. **Base de dados Restaurante** - base de dados contendo dados de restaurantes com 864 registros, sendo que 112 destes são duplicados. Cada registro possui 5 atributos e não há valores de atributos faltantes;

2. **Base de dados CORA** - base de dados contendo dados de citações de artigos científicos com 1295 registros, sendo que 1183 são duplicados. Cada registro possui 13 atributos e há vários valores de atributos faltantes;

O critério de escolha de nós no processo de treinamento da árvore de decisão é o índice de Gini [3] e os parâmetros do GP são:

1. **Tamanho da população:** 100;
2. **Critério de parada:** *fitness* igual a 1.0 ou 10 gerações;
3. **Função de *fitness*:** escore F1;
4. **Operador de seleção:** roleta;
5. **Método de cruzamento**: troca de sub árvore aleatória;
6. **Taxa de reprodução**: 30%;
7. **Taxa de mutação:** 1%;
8. **Método de mutação:** composição (para árvores com apenas um nó) troca equivalente de um nó aleatório (para árvores com mais de um nó);
9. Estratégia **elitista**.

[3] Medida de desigualdade criada pelo estatístico e sociólogo italiano Corrado Gini em 1912. Pode ser usada na indução de árvores de decisão substituindo as medidas de entropia.

Os experimentos foram executados 30 vezes para cada base de dados. Todos os experimentos foram desenvolvidos utilizando a linguagem de programação Python 2.7, numpy 1.12 e scikit-learn 0.18.1 (para as árvores de decisão) e jellyfish 0.5.6 (pacote que implementa as funções de similaridade). A execução foi divida entre vários computadores diferentes, todos utilizando o sistema operacional Ubuntu Server Linux 16.04. Os resultados são apresentados em forma de média e o respectivo desvio padrão do *fitness* dentre todas as execuções e utiliza validação cruzada de 4-folds, com 3 folds para treinamento e 1 para validação com permutações aleatórias.

O primeiro experimento avalia a capacidade de classificação com apenas um atributo. O objetivo desse experimento, além de servir de patamar de avaliação para os demais experimentos, é mostrar a qualidade dos melhores atributos de cada base de dados. A Tabela 10 mostra o atributo que mais se destacou de cada base de dados, o ponto de corte para classificação e seu respectivo operador de desigualdade.

Tabela 10 – Melhores classificadores com apenas um atributo

Base	Árvore	P.C.	F1 (σ)
Sintético 1	$E_{<ssid,DL>}$	$<= 2.000$	0.989 (0.010)
Sintético 2	$E_{<suburb,JW>}$	$>= 0.867$	0.974 (0.013)
Sintético 3	$E_{<suburb,JW>}$	$>= 0.827$	0.961 (0.012)
Cora	$E_{<title,3-grams>}$	$<= 0.532$	0.957 (0.002)
Restaurantes	$E_{<phone,2-grams>}$	$<= 0.300$	0.978 (0.021)

Todas as bases de dados possuem pelo menos uma evidência que alcança um *fitness* maior que 0.95. Com esses resultados, vale a pena ressaltar algumas diferenças da abordagem deste trabalho com os demais. A primeira diferença baseia-se na escolha do melhor ponto de corte: ao passo que outros trabalhos utilizam configurações com pontos de corte fixos, a abordagem deste trabalho busca uma melhor configuração de pontos de corte. Outra particularidade deste trabalho é o uso de um conjunto maior de funções de similaridade aproximada. Em (CARVALHO et al., 2012), por exemplo, não é usado as funções de similaridade da família *n-gram*, como o 3-gram, responsável pelo bom resultado de variáveis textuais nas bases Cora e Restaurantes.

O segundo experimento avalia soluções criadas pelo GP com um ou mais nós e tem o objetivo de analisar a capacidade do GP de combinar atributos para melhorar os resultados encontrados. A Tabela 11 exibe os resultados obtidos no segundo experimento.

os resultados apresentados na Tabela 11 mostram que o uso de GP para classificação melhorou a média dos resultados em quatro das cinco bases de dados. Sendo o segundo experimento uma abordagem presente na literatura atual, uma comparação dos resultados obtidos neste trabalho é necessária para avaliação da eficiência da abordagem utilizada. A Tabela 12 compara os resultados do segundo experimento deste trabalho com outros trabalhos disponíveis na literatura.

Tabela 11 – Melhores classificadores com um ou mais atributos

Base	Árvore	P.C.	F1 (σ)
Sintético 1	$E_{<ssid,3-grams>} - E_{pn,Jaro}$	<= 0.015	0.998 (0.004)
Sintético 2	$E_{<suburb,2-grams>-E<bn,JW>}$	<= -0.286	0.989 (0.007)
Sintético 3	$E_{<ssid,LV>} * E_{pn,3-grams}$	<= 2.000	0.995 (0.004)
Cora	$E_{<title,3grams}$	<= 0.532	0.957 (0.002)
Restaurantes	$E_{<name,JW} + E_{phone,JW}$	>= 1.698	0.987 (0.010)

Tabela 12 – Comparação do segundo experimento com (CARVALHO et al., 2012). Os valores apresentados da segunda coluna foram extraídos diretamente do trabalho do autor.

Base de dados	F1 (σ) - (CARVALHO et al., 2012)	F1(σ) - abordagem atual
Sintético 1	0.983 (0.032)	**0.998 (0.004)**
Sintético 2	0.978 (0.025)	**0.989 (0.007)**
Sintético 3	0.978 (0.025)	**0.995 (0.004)**
Cora	0.910 (0.010)	**0.957 (0.002)**
Restaurantes	0.980 (0.010)	**0.987 (0.010)**

O terceiro experimento tem como objetivo avaliar as possíveis melhorias que as árvores de decisão podem trazer para a abordagem de atributos únicos e servir de patamar de comparação para o terceiro experimento com o (CARVALHO et al., 2012). A Tabela 13 mostra a comparação dos resultados dos dois primeiros experimentos com os resultados do terceiro experimento.

Tabela 13 – Comparação dos experimentos realizados.

Base	(1) F1 (σ)	(2) F1(σ)	(3) F1 (σ)	F1 (σ) [4]
Sintético 1	0.989 (0.010)	0.998 (0.004)	**0.999 (0.003)**	0.983 (0.032)
Sintético 2	0.974 (0.013)	0.989 (0.007)	**0.995 (0.006)**	0.978 (0.025)
Sintético 3	0.961 (0.012)	0.995 (0.004)	**0.998 (0.002)**	0.978 (0.025)
Cora	0.957 (0.002)	0.957 (0.002)	**0.984 (0.004)**	0.910 (0.010)
Restaurante	0.978 (0.021)	0.987 (0.010)	**0.996 (0.008)**	0.980 (0.010)

Com os resultados em perspectiva, é possível notar a sensível melhora sobre os resultados anteriores em relação ao trabalho de (CARVALHO et al., 2012), o que evidencia algumas características práticas da abordagem seguida neste trabalho:

1. Primeiro, o uso de técnicas mais elaboradas para obtenção de um ponto de corte tem condições de melhorar significativamente o poder de classificação de um único atributo;

2. Segundo, mesmo com resultados relativamente bons de classificadores de apenas um atributo, o GP consegue combinar diferentes atributos e funções de similaridade para um resultado ligeiramente melhor;

3. Terceiro, quando o GP não é suficiente para encontrar um resultado melhor em relação ao simples uso de um único atributo para classificação, o uso de árvores de decisão potencializa os atributos descobertos pelo GP através da combinação dos mesmos para um melhor resultado;

4. E, por último, o potencial de melhoria por uma árvore de decisão mostra-se significativo em todos os cenários.

A discussão dos resultados aqui demonstrados frente aos objetivos originais deste trabalho torna-se necessária para avaliação do grau de sucesso obtido da abordagem descrita em capítulos anteriores. Portanto, o próximo capítulo expõe uma discussão sobre os resultados aqui demonstrados em face aos objetivos originais deste trabalho.

5 Conclusões e trabalhos futuros

Este trabalho teve como objetivo principal a busca de formas de geração de regras de ligação mais eficientes do ponto de vista de eficácia (*fitness*) para deduplicação de registros utilizando programação genética e indução de árvores de decisão. Tendo como argumento os resultados da metodologia utilizada foram apresentados no Capítulo 4, é possível dizer que este trabalho alcançou o objetivo proposto.

Três novas contribuições para melhorar o processo de deduplicação são expostas neste trabalho, com melhores resultados em relação à abordagem apresentada em (CARVALHO et al., 2012). A primeira delas evidencia que uma escolha mais elaborada de pontos de corte melhora consideravelmente o bom uso de atributos para classificação das bases de dados. A segunda permite que a GP inicie com uma população mais adequada para tratar o problema. A terceira demonstra o potencial de melhora nas classificações utilizando árvores de decisão com novos atributos construídos utilizando uma abordagem baseada em programação genética.

As contribuições aqui apresentadas oferecem uma melhora real do tratamento do problema de deduplicação de registros e, nos limites do nosso conhecimento, são contribuições inéditas em relação a literatura científica sobre o tema.

Em relação a trabalhos futuros, é necessário avaliar a viabilidade da abordagem aqui apresentada em bases de dados maiores dimensões, tanto em termos de atributos quanto em termos de registros, pois a maior base de dados que foi testada neste trabalho não possui mais de 1300 registros e duas dezenas de atributos. Devido ao alto custo computacional da solução, a abordagem aqui apresentada pode não ser satisfatória para vários casos de uso e, por isso, uma investigação mais aprofundada da escalabilidade do método é necessária.

Referências

CARVALHO, M. G. de et al. A genetic programming approach to record deduplication. *IEEE Transactions on Knowledge and Data Engineering*, v. 24, n. 3, p. 399–412, March 2012. ISSN 1041-4347. , ii, 1, 2, 3, 6, 8, 11, 12, 22, 25, 27, 28, 30, 31, 33

CHRISTEN, P. Febrl – an open source data cleaning, deduplication and record linkage system with a graphical user interface (demonstration session. In: *In ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'08*. [S.l.: s.n.], 2008. p. 1065–1068. 28

CHRISTEN, P. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, v. 24, n. 9, p. 1537–1555, Sept 2012. ISSN 1041-4347. i, 1, 4, 5, 6

DAMERAU, F. J. A technique for computer detection and correction of spelling errors. *Commun. ACM*, ACM, New York, NY, USA, v. 7, n. 3, p. 171–176, mar. 1964. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/363958.363994>. 7

DEHGHANPOUR, S.; ESFAHANIPOUR, A. A robust genetic programming model for a dynamic portfolio insurance strategy. In: *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. [S.l.: s.n.], 2017. p. 201–206. 11

ELMAGARMID, A. K.; IPEIROTIS, P. G.; VERYKIOS, V. S. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, v. 19, n. 1, p. 1–16, Jan 2007. ISSN 1041-4347. 1

ESPEJO, P. G.; VENTURA, S.; HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 40, n. 2, p. 121–144, March 2010. ISSN 1094-6977. 11

FAYYAD, U. M.; IRANI, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In: *IJCAI 1993*. [S.l.]: AAAI Press, 1993. p. 1022–1027. 19, 26

FELLEGI, I. P.; SUNTER, A. B. A Theory for Record Linkage. *Journal of the American Statistical Association*, Taylor & Francis, v. 64, n. 328, p. 1183–1210, dez. 1969. Disponível em: <http://dx.doi.org/10.1080/01621459.1969.10501049>. , 1

FENTON, M. et al. Multilayer optimization of heterogeneous networks using grammatical genetic programming. *IEEE Transactions on Cybernetics*, v. 47, n. 9, p. 2938–2950, Sept 2017. ISSN 2168-2267. 11

FREITAS, J. de et al. Active learning genetic programming for record deduplication. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8. ISSN 1089-778X. 1

HAMMING, R. Error Detecting and Error Correcting Codes. *Bell System Techincal Journal*, v. 29, p. 147–160, 1950. 7

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. [S.l.]: The University of Michigan Press, 1975. 9

ISELE, R.; BIZER, C. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, v. 5, n. 11, p. 1638–1649, Aug 2012. 1

ISELE, R.; BIZER, C. Active learning of expressive linkage rules using genetic programming. *Web Semantics: Science, Services and Agents on the World Wide Web*, v. 23, p. 2 – 15, 2013. ISSN 1570-8268. Data Linking. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1570826813000231>. 1

JARO, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, v. 84, n. 406, p. 414–420, 1989. 7

KOZA, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, v. 4, n. 2, p. 87–112, 1994. ISSN 1573-1375. Disponível em: <http://dx.doi.org/10.1007/BF00175355>. 10, 11, 12

LEVENSHTEIN, V. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, v. 10, p. 707, 1966. 7

MELANIE, M. *An Introduction To Genetic Algorithms*. [S.l.]: The MIT Press, 1999. ISBN 0262133164. 10

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072. 14, 15, 26

MRUGALA, K.; TUPTUK, N.; HAILES, S. Evolving attackers against wireless sensor networks using genetic programming. *IET Wireless Sensor Systems*, v. 7, n. 4, p. 113–122, 2017. ISSN 2043-6386. 11

MUHARRAM, M.; SMITH, G. D. Evolutionary constructive induction. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 11, p. 1518–1528, Nov 2005. ISSN 1041-4347. 2, 13, 19, 20

NGOMO, N.; LYKO, K. Eagle: Efficient active learning of link specifications using genetic programming. In: _____. *The Semantic Web: Research and Applications: 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 149–163. ISBN 978-3-642-30284-8. Disponível em: <https://doi.org/10.1007/978-3-642-30284-8_17>. 1

QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1022643204877>. 13

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, July 1948. ISSN 0005-8580. 7

SUN, C. et al. A genetic algorithm based entity resolution approach with active learning. *Frontiers of Computer Science*, v. 11, n. 1, p. 147–159, Feb 2017. ISSN 2095-2236. Disponível em: <https://doi.org/10.1007/s11704-015-5276-6>. 1

TAHIR, N. *Fuzzy string matching using cosine similarity*. 2015. <https://blog.nishtahir.com/2015/09/19/fuzzy-string-matching-using-cosine-similarity/>. Acessado: 2017-11-30. 7

TAN, P. *Introduction To Data Mining*. Pearson Education, 2006. ISBN 9788131714720. Disponível em: <https://books.google.com.br/books?id=Wx4NPK4qHXsC>. 15

WINKLER, W. E. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: *Proceedings of the Section on Survey Research*. [S.l.: s.n.], 1990. p. 354–359. 7