Uma avaliação de Eficiência e Eficácia da Combinação de Técnicas para Deduplicação de Dados

Levy de Souza Silva[1]*, Dimas Cassimiro Nascimento Filho[2], Mirella M. Moro[1]

[1]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901 – Brasil

[2]Universidade Federal Rural de Pernambuco (UFRPE)
Unidade Acadêmica de Garanhuns (UAG) - Garanhuns - PE - Brasil

{levysouza,mirella}@dcc.ufmg.br, dimascnf@uag.ufrpe.br

Abstract. *Data Deduplication is the task of identifying and eliminating duplicate records in a single database. It is a complex process that involves several steps, including: defining blocking key, similarity function and indexing method. There are several approaches for each of these steps. In this context, the objective of this work is to find the best combination for such algorithms aiming to improve the efficiency and effectiveness of the deduplication process as a whole. To this end, we present an experimental evaluation using real and artificial datasets. The results point to distinct combinations that present better results in specific situations.*

Resumo. *Deduplicação de dados é a tarefa de identificar e eliminar registros duplicados em um única base de dados. É um processo complexo que envolve várias etapas, incluindo: definição de chave de bloco, função de similaridade e método de indexação. Existem diversas abordagens para cada uma dessas etapas. Então, o objetivo deste trabalho é encontrar a melhor combinação para tais algoritmos visando melhorar a eficiência e eficácia do processo como um todo. Para tal, apresentamos uma avaliação experimental utilizando bases de dados reais e artificiais. Os resultados apontam para combinações distintas que apresentam melhor resultados em situações específicas.*

1. Introdução

Devido a enorme quantidade de dados coletados e armazenados ao longo dos anos pelas empresas, agências de governo e projetos de pesquisa, o estudo de técnicas que permitam a análise, processamento e mineração dos dados de forma eficiente têm sido atraído pela acadêmia e indústria. Uma tarefa que tem ganhado relevância em muitos domínios de aplicação é a tarefa de deduplicação de dados. Deduplicação de dados é uma abordagem para identificar e eliminar registros duplicados em bases de dados. Registros duplicados são instâncias de dados que representam a mesma entidade no mundo real. Esta tarefa é referenciada por diversas nomenclaturas na literatura, incluindo: I) *Data Deduplication* e *Duplicate Detection*, quando o processo ocorre em uma única fonte de dados; e II) *Record Linkage*, *Entity Resolution*, *Data Matching* e *Object Identification*, quando são utilizadas várias fontes de dados [Christen 2012a].

*Trabalho desenvolvido enquanto estudava na UFRPE

No setor da saúde, em sistemas de vigilância, por exemplo, a deduplicação de dados tem o objetivo de detectar padrões suspeitos, tais como surtos de doenças contagiosas. Muitas empresas usam essa abordagem para melhorar a qualidade dos dados, compilar listas de discussão ou para comparar dados entre organizações. Órgãos do governo estão empregando cada vez mais as técnicas de deduplicação em serviços de segurança social buscando identificar pessoas que se inscreveram várias vezes para os programas sociais. Outra área de aplicação que tem ganhado interesse está relacionada com a detecção de crimes e fraudes [Christen 2012a]. Agências de segurança e investigadores de crimes buscam rapidamente fazer o cruzamento dos dados de criminosos, presentes em várias fontes, com o intuito de encontrar indivíduos que representam a mesma pessoa mas fornecem informações pessoais fictícias quando questionados em depoimentos.

O problema de encontrar registros duplicados não é aplicado somente a dados pessoais. Às vezes é preciso identificar duplicidade em registros de empresas, produtos de consumidores, publicações, citações bibliográficas, páginas web, resultados de sistemas de busca ou até mesmo sequências de genoma. No campo de recuperação de informação, por exemplo, é importante remover documentos duplicados (páginas web e citações bibliográficas) dos resultados retornados pelos motores de busca, pelas bibliotecas digitais ou até mesmo pelos sistemas automáticos de indexação de texto [Hajishirzi et al. 2010]. Outra aplicação com crescimento significante é a identificação de produtos duplicados em sistemas de lojas online [Christen 2012a].

A identificação de registros duplicados é um processo complexo e composto de várias epatas. A Figura 1 apresenta uma visão geral desta tarefa em uma abordagem de *Record Linkage*. O passo a passo é composto das seguintes etapas: I) o conjunto dos dados é submetido a um pré-processamento para deixar os dados mais limpos e concisos; II) todos os registros precisam ser indexados por uma chave de bloco (BK) com o intuito de criar pequenos grupos de registros que podem corresponder ao mesmo registro, evitando assim uma comparação produto cartesiano entre os dados; III) todos os registros pertencentes a cada um dos blocos são comparados por meio de funções de similaridade; IV) os registros são classificados como duplicados, não duplicados e como possíveis duplicados de acordo com um limiar de similaridade; e V) todo o processo é avaliado com o intuito de medir a eficácia e eficiência dos algoritmos aplicados em cada uma das etapas. Para identificar registros duplicados em uma única base de dados, como acontece na abordagem de *Data Deduplication*, o processo exibido na Figura 1 também pode ser aplicado, porém apenas uma fonte de dados é utilizada para prover os dados.

Nos últimos anos, vários estudos de deduplicação em diferentes contextos foram publicados [Borges et al. 2008, Canalle et al. 2016, Chen et al. 2012, Christen 2012b, Machado et al. 2016, Vesdapunt et al. 2014]. Assim, embora tenham sido desenvolvidas várias técnicas para cada uma das etapas do processo, uma pequena quantidade de estudos apresentam um levantamento experimental incluindo tais técnicas. Além disso, poucos estudos abordam métodos que buscam analisar, no mesmo ambiente experimental, quais os melhores algoritmos a serem utilizados em cada uma das etapas do processo.

Nesse sentido, este trabalho tem o objetivo de avaliar a eficiência e eficácia da combinação de algoritmos, técnicas e métodos de deduplicação de dados. Para isso, diversos experimentos foram executados em um conjunto de bases de dados reais e artificiais, com tamanho variando entre 10.000 e 270.00 instâncias, contendo registros originais

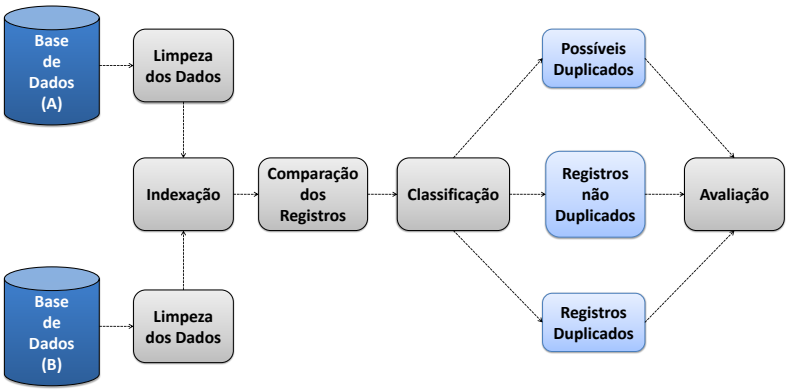


Figura 1. Processo universal de identificação de registros duplicados em uma abordagem de *Record Linkage*. Adaptado de [Christen 2012a].

e duplicados. Dessa forma, as principais contribuições deste trabalho são: I) realizar uma avaliação de eficiência entre as funções de similaridade, métodos de indexação e chaves de bloco; II) realizar uma avaliação de eficácia diante da combinação das técnicas aplicadas; e III) propor uma melhor combinação de algoritmos para serem utilizados nas etapas da deduplicação de dados.

As próximas seções estão organizadas da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 aborda a fundamentação teórica. A Seção 4 expõe os experimentos realizados, as discussões e os resultados alcançados. A Seção 5 apresenta as conclusões do trabalho e aponta trabalhos futuros.

2. Trabalhos Relacionados

Machado et al. [2016] apresentam um método para deduplicação de contatos de dispositivos móveis utilizando similaridade textual e aprendizagem de máquina. Os escores produzidos por meio das funções de similaridade são destinados a treinar um modelo de classificação que pode ser adotado como estratégia para integração dos contatos duplicados. A qualidade do método é avaliada em uma base de dados real com 2.000 contatos.

Canalle et al. [2016] apresentam uma abordagem de seleção de atributos mais relevantes para a etapa de comparação dos registros em um processo de integração de dados de bibliotecas digitais. Diversos critérios são extraídos dos dados com o intuito de identificar a relevância de cada atributo. Para avaliar a relevância, são consideradas métricas que calculam a repetição e densidade dos atributos, por exemplo, e métricas relacionadas a qualidade das fontes dos dados.

Vesdapunt et al. [2014] modelam o problema da identificação de registros duplicados como um grafo. Cada aresta denota a probabilidade do par de registros (vértices) representarem o mesmo objeto. O objetivo do trabalho é abordar o problema em um contexto de *crowdsourcing* e utilizar o conhecimento prévio de comparações para evitar comparações futuras, melhorando dessa forma o desempenho dos algoritmos. Por exemplo, considerando que $a = b$ e que $b = c$, pode ser inferido que $a = c$?

Christen [2012b] apresenta um *survey* de métodos de indexação e realiza uma avaliação experimental com os métodos apresentados. No trabalho foram analisados os

métodos *Standard Blocking*, *Sorted Neighborhood Indexing*, *Q-Gram-Based Indexing*, *Suffix Array-Based Indexing*, *Canopy Clustering* e *String-Map-Based Indexing*. O objetivo do trabalho foi avaliar a eficiência das técnicas a fim de responder a perguntas como: De que maneira os valores das chaves de bloco geradas influenciam na qualidade dos pares de registros candidatos criados? Como os diferentes métodos se comportam em diversos tipos de dados? Quais as práticas que se mostram com melhor escalabilidade às bases de dados de maior dimensão? Segundo o autor, a técnica *Q-gram Based Indexing* apresenta-se como uma das mais lentas, não sendo apropriada para deduplicação de grandes bases de dados. Por outro lado, ele afirma que as abordagens mais tradicionais, como a definição de blocos e a metodologia de vizinhança são as técnicas mais rápidas.

Chen et al. [2012] propoem um método baseado em aprendizagem de máquina para selecionar o atributo ou o grupo de atributos mais relevantes para a etapa de comparação dos registros. Além disso, o limiar e os pesos de similaridade são definidos automaticamente. Funções de similaridade como *Jaccard Distance*, *Jaro Distance* e *Edit Distance* foram utilizadas para comparação das instâncias e experimentos foram executados em uma portação da base de dados do *CORA*.

Borges et al. [2008] apresentam uma abordagem para identificação de registros duplicados em um contexto de bibliotecas digitais visando integrar dados de várias fontes. São especificadas diversas funções de similaridade, incluindo: *YearSim*, *IniSim*, *NameMatch* e *MetadataMatch*. Segundo os autores, essas funções identificam com maior eficácia os registros duplicados em um conjunto de metadados de objetos digitais e podem ser utilizadas para a etapa de comparação das instâncias.

Desse modo, observa-se que nenhum dos estudos citados, com exceção de Christen [2012b], apresenta uma metodologia para analisar ao mesmo tempo os melhores algoritmos a serem aplicados em cada uma das etapas do processo. Considerando o trabalho de Christen [2012b] que investigou métodos de indexação e seus comportamentos em diversos contextos de base de dados, o objetivo aqui não é investigar apenas os métodos de indexação, mas analisar no mesmo plano de experimentos todo o conjunto de algoritmos que são aplicados em cada uma das etapas do processo de deduplicação de dados.

3. Fundamentação Teórica

Duplicatas. Um registro duplicado ou simplesmente uma duplicata, no contexto de banco de dados, se refere a um registro gerado com base em uma entidade original, que sofreu algumas modificações ou até mesmo registros que são produzidos por meio de erros causados pelo usuário durante o processo de digitação.

3.1. Funções de Similaridade.

Diversas funções de similaridade são propostas pela literatura para comparação de registros. De forma geral, estas funções têm o objetivo de calcular computacionalmente o quanto dois valores são correspondentes, ou seja, calcular um valor de similaridade no intervalo $[0, 1]$, entre dois registros. Assim, o valor obtido é igual a 1 quando há concordância total entre os registros comparados.

***Jaro*.** Tem o objetivo de verificar o número de caracteres iguais em dois registros [Jaro 1989]. Essa função leva em conta o tamanho das variáveis e ainda os tipos de

erros que comumente ocorrem com variáveis alfanuméricas. Dessa forma, um caractere é considerado em comum se estiver presente em ambos os registros ao mesmo tempo. Além disso, o número de transposições realizadas para se obter a igualdade entre dois caracteres é observado. Assim, para dois registros a e b, uma transposição ocorrerá quando dois valores que ocupem a mesma posição forem diferentes, ou seja, $a[i] \neq b[i]$. A função estabelece ainda que a distância entre os itens permutados não ultrapasse a metade do comprimento do menor registro.

Jaro Winkler. É uma extensão do *Jaro* e foi proposto por Winkler [1990]. Essa função também investiga a similaridade, porém observa de forma mais precisa a igualdade dos quatro primeiros caracteres, ou seja, existe um número $(0 \leq p \leq 4)$, que representa o número de elementos em comum nas i primeiras instâncias dos registros a serem comparados. Por exemplo, para o par de valores "$peter$" e "$petra$", existe um prefixo comum "pet", determinando assim $p = 3$.

***Levenshtein* ou *Edit Distance*.** É uma das primeiras funções utilizadas na comparação de registros. Considerando dois registros X e Y, é calculado o número mínimo de modificações necessárias para transformar X em Y, ou vice versa. De acordo com Navarro [2001], essas alterações restringem-se às funções de inserção, remoção ou substituição de caracteres. Sendo assim, para transformar por exemplo, o valor "$kitten$" em "$sitting$" são necessárias no mínimo três mudanças: I) *sitten* - (substituição de k por s); II) *sittin* - (substituição de e por i); e III) *sitting* - (inserção de g no final).

3.2. Chaves de Indexação.

O gargalo de desempenho em um sistema de deduplicação de dados é geralmente determinado pelo número de comparações entre os registros, o que pode ser auxiliado por meio de um bom índice. Segundo Christem [2012b], a indexação pode ser vista como uma filtragem, porque se baseia em criar estruturas de BKs que agrupam valores semelhantes.

***Soundex*.** É uma técnica desenvolvida e patenteada por Odell e Russell [1918], com base na pronúncia da língua inglesa. O *soundex* codifica uma sequência de caracteres mantendo o primeiro caractere e converte os demais para dígitos entre zero e seis. A Tabela 1 apresenta um exemplo dessa codificação.

***Suffix*.** Utiliza uma metodologia que cria *substrings* de uma palavra, chamadas de sufixos. Segundo Christen [2012a], os sufixos de uma *string* são todas as suas *substrings* com um ou mais caracteres. A Tabela 3 apresenta exemplos de sufixos.

3.3. Método de Indexação.

Diversos métodos são propostos para comparação dos registros. Os principais métodos utilizam conceitos de comparação por blocos e por vizinhos.

Tabela 1. *Soundex* [Christen 2012a].

Letra	Dígito
a, e, i, o, u, y, h, w	0
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Tabela 2. Exemplos de Codificação *Soundex*.

Nome	Código *Soundex*
peter	p360
pete	p300
pedro	p360
stephen	s315
steve	s310
smith	s530
smythe	s530

Tabela 3. Exemplos de Sufixos de uma *String* Apresentados por Christen [2012a].

ID	Nome	Sufixos
r1	katherina	katherina, atherina, therina, herina, erina, rina
r2	catrina	catrina, atrina, trina, rina
r3	catherine	catherine, atherine, therine, herine, erine, rine

Tabela 4. Comparação de Registros com *Standard Blocking* [Christen 2012a].

RecID	GivenName	Surname	Sndx(GiN)	Sndx(SurN)
a1	peter	myler	p360	**m460**
a2	pedro	smith	p360	s530
a3	steve	peters	s315	p362
a4	christine	miller	c623	**m460**
b1	kristina	miller	k623	**m460**
b2	stephen	peter	s315	p360
b3	kylie	smith	k400	s530
b4	pete	myler	p300	**m460**

***Standard Blocking*.** É um algoritmo que tem o intuito de gerar um conjunto de blocos, onde cada bloco agrupa entidades semelhantes. Logo, apenas as entidades que pertencem ao mesmo bloco são comparadas entre si. Esses agrupamentos são definidos de acordo com as BKs, geradas para cada instância. Segundo Fellegi e Sunter [1969], essa é uma das técnicas mais utilizadas durante décadas, abordando métodos para correspondência de dados e para diminuição das redundâncias. A Tabela 4 apresenta um exemplo de comparação dos elementos utilizando a técnica *Standard Blocking*. Considerando a chave de bloco **m460** da coluna *Sndx(SurN)*, por exemplo, as comparações são realizadas com os valores: (a1,a5), (a1,b1), (a1,b4), (a5,b1), (a5,b4) e (b1,b4).

***Sorted Neighborhood*.** É uma alternativa para definição de quais registros comparar em um conjunto de dados. Esse conceito foi apresentado por Hernández e Stolfo [1995] , onde ao invés de definir vários blocos, criados por meio da indexação, os valores são combinados através de uma chave de triagem, que é semelhante a uma BK. Dessa forma, antes de realizar as comparações em um banco de dados, todos os elementos são ordenados de acordo com a BK definida para cada elemento. Após isso, uma janela deslizante de tamanho $w > 1$ percorre todas as instâncias do banco de dados e, enquanto isso, o primeiro valor da janela é comparado com todos os outros.

4. Experimentos e Discussões

A metodologia dos experimentos[1] consiste em realizar uma combinação das técnicas aplicadas na deduplicação de dados, ou seja, para cada tratamento foi utilizado: uma chave de bloco, uma função de similaridade e um método de indexação. Nas experimentações relacionadas a eficácia, foram avaliadas medidas de *Precision*, *Recall* e *F-Measure* [Brizan and Tansel 2006]. A avaliação de eficácia ateve-se a verificar qual o tratamento que, quando aplicado em uma única base de dados, obteve o melhor valor de *F-Measure*, relacionando este ao total de duplicatas detectadas.

Para definir o limiar de similaridade e o tamanho w das janelas utilizadas no algoritmo de vizinhança foram realizados testes em algumas bases de dados. Assim, investigou-se com quais valores a *F-Measure* foi melhorada. Deste modo, após os testes iniciais, estabeleceu-se um limiar de 0.90 para a similaridade e o tamanho w foi definido

[1]Disponível em: https://www.dropbox.com/s/49xpytfl3ldkmms/C%C3%B3digoFonte.zip?dl=0

Tabela 5. Plano de Experimentos Realizados com a Técnica *Standard Blocking*.

Trat.	Indexação	Chave	Função
1	SB	*Soundex*	*JaroWinkler*
2	SB	*Suffix*	*JaroWinkler*
3	SB	Ambos	*JaroWinkler*
4	SB	*Soundex*	*Jaro*
5	SB	*Suffix*	*Jaro*
6	SB	Ambos	*Jaro*
7	SB	*Soundex*	*Edit Distance*
8	SB	*Suffix*	*Edit Distance*
9	SB	Ambos	*Edit Distance*

Tabela 6. Plano de Experimentos Realizados com a Técnica *Sorted Neighborhood*.

Trat.	Indexação	Chave	Função
10	SN	*Soundex*	*JaroWinkler*
11	SN	*Suffix*	*JaroWinkler*
12	SN	Ambos	*JaroWinkler*
13	SN	*Soundex*	*Jaro*
14	SN	*Suffix*	*Jaro*
15	SN	Ambos	*Jaro*
16	SN	*Soundex*	*Edit Distance*
17	SN	*Suffix*	*Edit Distance*
18	SN	Ambos	*Edit Distance*

de acordo com a quantidade de tuplas na base de dados. Logo, w assumiu os valores: I) $w = 50$, para as bases de dados com menos de 10.000 instâncias; II) $w = 150$, para as bases de dados com mais de 10.000 e menos de 50.000 instâncias; e III) $w = 500$, para as bases de dados maiores.

A avaliação da eficiência consiste em calcular o tempo médio de 30 execuções em cada tratamento de cada base de dados. Os experimentos executaram-se em um *notebook* com processador *Pentium Dual-Core 2.30GHz*, memória *RAM* de 2,00GB e Sistema Operacional *Windows* 7 de 32 bits. As Tabelas 5 e 6 apresentam os tratamentos empregados no experimento, esses tratamentos foram executados em todas as bases de dados.

Bases de Dados[2]. O conjunto dos dados utilizados nos experimentos foram extraídos de fontes livres e artificiais, formando um total de seis bases de dados. Assim, três delas procederam de dados livres, uma foi criada pela ferramenta *Data Set Generator Program*[3], uma foi baixada do endereço Dados Prontos[4] e uma foi selecionada da Superintendência de Seguros Privados[5](SUSEP).

O *Data Set Generator Program* é uma ferramenta que produz dados e duplicatas artificialmente de acordo com os parâmetros do usuário. Ela é parte do projeto *Febrl* [6] e é baseada nas ideias apresentadas por [Hernández and Stolfo 1995]. Os dados livres estão disponíveis no Portal Brasileiro de Dados Abertos[7]. O SUSEP contém dados extraídos dos formulários de informações periódicas enviados pelas companhias seguradoras, resseguradoras, entidades abertas de previdência privada e sociedades de capitalização.

Geração das Duplicatas. Com exceção da base de dados completa gerada pela ferramenta Febrl, todas as duplicatas foram produzidas por meio da ferramenta UAG-DUP[8]. UAG-DUP é uma ferramenta para produção de duplicatas utilizando bases de dados reais e parâmetros customizáveis. Ela foi desenvolvida, durante o trabalho, em vista da dificuldade encontrada no *Data Set Generator Program*, que não possibilita a concepção de dados duplicados utilizando uma base de dados específica. A UAG-DUP adota os principais conceitos para produção de duplicatas apresentados por [Ioannou et al. 2013].

[2]Disponível em: https://www.dropbox.com/s/upashltf5pmul6t/BasesDeDadosExperimentos.sql?dl=0
[3]Disponível em: https://cs.anu.edu.au/people/Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/node70.html
[4]Disponível em: https://gabrielrb.net/2011/10/18/dados-prontos-em-formato-sql-e-csv/
[5]Disponível em: https://www2.susep.gov.br/menuestatistica/SES/principal.aspx
[6]Disponível em: https://sourceforge.net/projects/febrl/
[7]Disponível em: www.dados.gov.br
[8]Disponível em: https://www.dropbox.com/s/56w9cgwyl6wpf49/UAG-DUP.zip?dl=0

Tabela 7. Proporção de Duplicatas e Registros Originais nas Bases de Dados.

Nº	Nome	Nº de Reg.	Tamanho	% de Duplicatas	QTDMAXDPI	NMAXMODPR
1	UBS5	38.875	$4,41MB$	5%	3	2
2	UBS10	40.729	$4,63MB$	10%	5	3
3	BDT5	10.132	$567KB$	5%	3	2
4	BDT10	10.614	$958KB$	10%	5	2
5	RC5	48.059	$8,44MB$	5%	3	3
6	RC10	50.350	$8,86MB$	10%	5	3
7	RP5	10.208	$1,49MB$	5%	3	3
8	RP10	10.697	$1,57MB$	10%	5	2
9	SUSEP10	270.146	$31,2MB$	10%	3	3
10	FebrlData	55.001	$9,20MB$	10%	4	4
11	NFebrl	58.126	$3,15MB$	10%	4	2

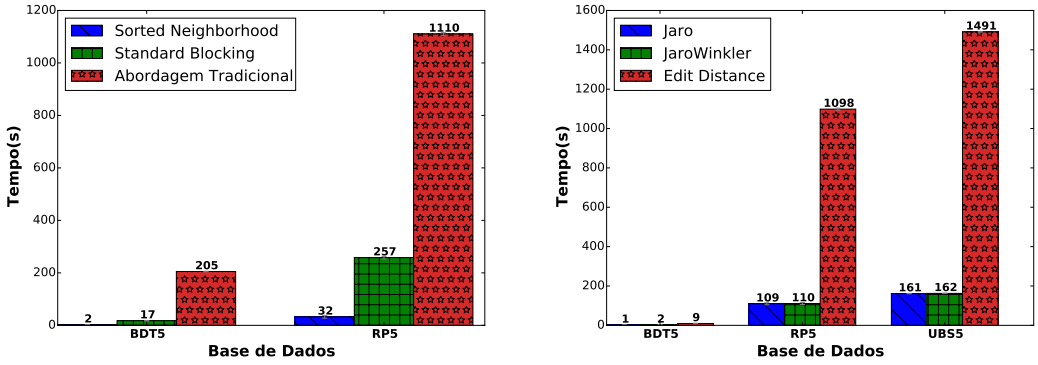
Assim, para cada uma das bases de dados selecionadas criaram-se duas cópias, sendo a primeira com uma porcentagem de 5% de registros duplicados e a outra com 10% de duplicatas. Para a base de dados dos administradores da SUSEP e a dos nomes concebidos pelo *Data Set Generator Program*, foi produzido um percentual de 10% de duplicatas. A Tabela 7 apresenta: I) as informações das bases de dados; II) a quantidade máxima de duplicatas geradas de cada instância (QTDMAXDPI); e III) o número máximo de modificações em cada registro (NMAXMODPR) para geração de uma duplicata.

4.1. Avaliação da Eficiência

A Figura 2a apresenta uma comparação dos tempos de execução dos algoritmos *Standard Blocking* e *Sorted Neighborhood* em relação ao tempo consumido quando não utilizou-se nenhuma das técnicas, ou seja, cada elemento da base de dados foi comparado com todos os outros. Percebe-se, que a ausência de estratégias de indexação contribui para um tempo de execução elevado. Além disso, os tempos de execução possuem variações consideráveis de acordo com a quantidade de tuplas existentes no banco de dados, como observado nas bases $BDT5$ e $RP5$. Sendo assim, a estratégia de vizinhança, por exemplo, é 34 vezes mais rápida do que o método tradicional, como observa-se no conjunto de dados $RP5$. Esse tempo de execução superior é ocasionado pelo número total de pares candidatos que são criados, chegando, por exemplo, a um valor de $51.323.646$ na base de dados $BDT5$. Por sua vez, com o método de vizinhança, são criados 485.160 pares.

A Figura 2b exibe o tempo consumido pelas funções de similaridade para calcular a semelhança entre dois registros. Observa-se, que o tempo para o método *Edit Distance* é bastante elevado, chegando este a ser nove vezes mais lento do que o algoritmo *Jaro Winkler*, como é notado em $UBS5$. Sendo assim, é observado que a técnica *Edit Distance* não apresenta bons resultados quando o tamanho das *strings* que são comparadas são grandes, o que ocorre na base de dados $UBS5$. Por outro lado, quando o tamanho das *strings* é relativamente pequeno, como na base de dados $BT5$, o algoritmo apresenta um tempo de execução semelhante aos demais. Para as funções *Jaro* e *JaroWinkler*, nas bases de dados $BTD5$, $RC5$ e $UBS5$, os tempos de execução são semelhantes e as diferenças são quase imperceptíveis como exibido na Figura 2b.

A Figura 3 apresenta uma comparação para avaliar o desempenho do algoritmo *Standard Blocking*, de acordo com cada BKV definida. Percebe-se, que a eficiência do método está proporcionalmente relacionada a distribuição das BKs, ou seja, o número total de chaves distintas que são criadas influencia no tempo de execução do algoritmo.



(a) (b)

Figura 2. Tempo de Execução *Standard Blocking*, *Sorted Neighborhood* e Abordagem Tradicional (a), (b) Tempo de Execução das Funções de Similaridade.

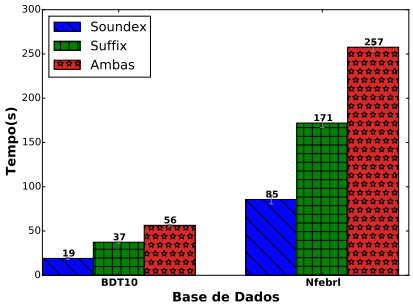


Figura 3. Tempo de Execução do Algoritmo *Standard Blocking*.

Neste exemplo, são comparadas as técnicas de indexação *soundex* e *suffix*, como também, a aplicação de ambos os métodos de indexação ao mesmo tempo. Sendo assim, foi observado que na base de dados $BDT10$, com a técnica *soundex*, criaram-se 2.437 blocos e conceberam-se 87.271 pares candidatos, por outro lado, com a *suffix*, originaram-se 4.361 blocos e formaram-se 130.015 pares candidatos.

Diante disso, é observado que existe um ganho significativo no quesito eficiência quando certas técnicas de indexação e funções de similaridade são aplicadas. Considerando o trabalho de Machado et al. [2016], por exemplo, em que os contatos foram comparados todos com todos, a indexação seria uma estratégia fundamental a ser adotada, pois tanto a memória quanto o processamento disponível nos dispositivos móveis são limitados. Logo, não é uma abordagem viável realizar todas as comparações.

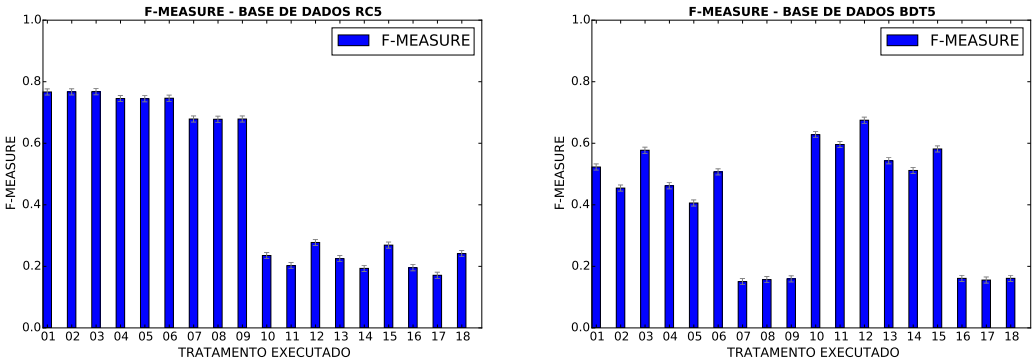
4.2. Avaliação da Eficácia

A Figura 4a apresenta os cálculos de *F-Measure* alcançados na base de dados $RC5$. Neste caso, o algoritmo *Jaro Winkler* produziu resultados melhores do que o *Jaro*, como visto no tratamento N^o03. Neste exemplo, a base de dados em questão apresenta uma média de 11 caracteres por *string* e a maioria dos seus dados figuram-se como *strings* do tipo: “Tim”, “Vivo”, “GVT”, “SKY”, “Banco BMG” e “Vivo - Telefônica”, o que torna mais fácil de existir uma concordância nas quatro primeiras instâncias dos registros duplicados criados, aumentando assim a similaridade *Jaro Winkler* em contrapartida a *Jaro*. Por outro lado, é

possível observar nesta base de dados que, comparando-se as técnicas de blocagem SB e SN, tratamentos de N^o01 à N^o09 e do N^o10 em diante, respectivamente, o algoritmo de vizinhança não obteve bons resultados. Este fato é ocasionado pela quantidade de tuplas existentes na base de dados. A base de dados $RC5$ possui cerca de 50.000 instâncias, o que pode distanciar o valor duplicado do original. Além disso, dependendo do tamanho w de janela que desliza sobre os dados a detecção das duplicatas fica comprometida, pois a janela pode não englobar todas as duplicatas.

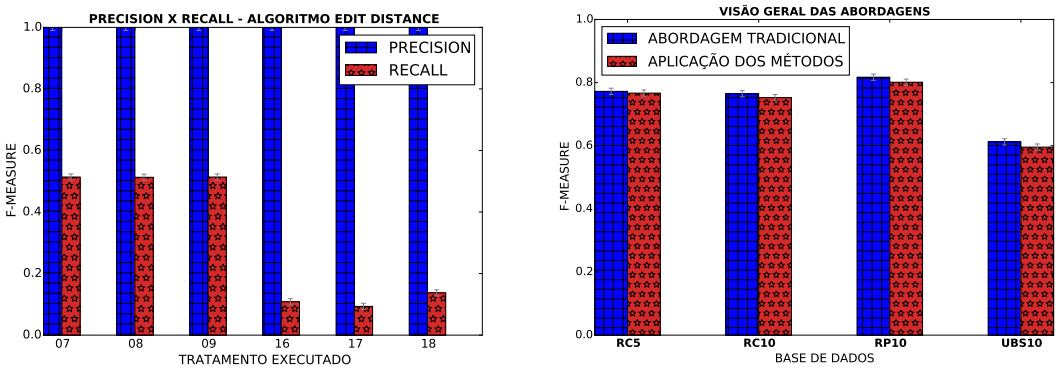
A Figura 4b exibe os cálculos de *F-Measure* obtidos na base de dados $BDT5$. É observado que o método de vizinhança obteve resultados mais significativos em quase todos os tratamentos, execuções N^o10 até a N^o18. E ainda, que a execução N^o12 na qual foi utilizada a função de similaridade *Jaro Winkler*, apresentou a melhor eficácia. Sendo assim, verificando-se a similaridade foi constatado que a base de dados utilizada apresenta *strings* pequenas nos seus atributos, abordando novamente a discussão exposta na Figura 4a. Dessa forma, atributos que contém uma pequena quantidade de caracteres, como os nomes de distritos, por exemplo, intensificam para as duplicatas a igualdade das 4 primeiras ocorrências dos caracteres. Por outro lado, quanto ao método de indexação, para a base de dados em questão, o resultado é atribuído também a quantidade de instâncias da mesma. Neste exemplo, a base de dados tem em média 10.000 tuplas. Sendo assim, os registros duplicados não se distanciam de forma drástica um do outro quando ocorre a indexação, o que não acontece em bases de dados maiores.

A Figura 5a apresenta uma comparação da *Precision* com o *Recall* com foco no algoritmo *Edit Distance* na base de dados $RC5$. Observa-se neste exemplo que, para os tratamentos N^o07, N^o08, N^o09, N^o16, N^o17 e N^o18 a função de similaridade *Edit Distance* apresentou uma precisão máxima, ou seja, não classificou nenhum registro que não era duplicado como um valor duplicado, mas por outro lado, o *recall* é bastante comprometido, onde identificou-se uma quantidade insignificante de duplicatas, o que piora drasticamente a medida de *F-Measure*. Sendo assim, os resultados ruins estão atrelados ao tamanho das *strings* de cada atributo, pois é necessário um maior número de modificações para transformar um valor em outro, o que diminuirá o valor de similaridade dos atributos comparados, comprometendo o *recall* e, consequentemente, a *F-Measure*.



(a) (b)

Figura 4. Eficácia: *Jaro* X *Jaro Winkler* na Base de Dados $RC5$ (a), (b) Eficácia dos Algoritmos SB e SN na Base de Dados $BTD5$.



(a) (b)

Figura 5. *Precision* X *Recall* Algoritmo *Edit Distance* (a), (b) Avaliação Geral das Técnicas de Blocagem, Funções de Similaridade e Método de Indexação.

A Figura 5b apresenta as medidas de *F-Measure* alcançadas em uma comparação onde foi utilizado no primeiro caso, a combinação dos algorítimos e, no segundo, a ausência dos métodos de indexação, ou seja, comparando-se todos os registros da base de dados. Dessa forma, aplicaram-se nas bases de dados $RC5$, $RC10$, $RP10$ e $UBS10$ o método de indexação *Standard Blocking*, ambas as chaves de bloco e a função de similaridade *JaroWinkler*. Assim, pode ser observado que com a aplicação da combinação dos métodos os resultados obtidos são significantes para o processo de deduplicação de dados, tanto para bases de dados com poucas instâncias quanto para bases de dados maiores.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou um estudo sobre a avaliação da eficiência e eficácia da combinação de técnicas para deduplicação de dados. Diferente dos trabalhos anteriores, a metodologia apresentada buscou comparar todas as técnicas no mesmo ambiente experimental. Diversos métodos e técnicas do estado da arte foram implementados e vários experimentos envolvendo bases de dados reais e artificiais foram executados.

Comparando-se os métodos de indexação, observou-se que para as base de dados com um conjunto pequeno de instâncias, como a $BTD5$, o método de vizinhança mostrou-se eficaz e muito mais eficiente do que o método *Standard Blocking*. Para as base de dados maiores, como a $RC5$, a utilização da técnica de blocos produziu uma eficácia melhor. Assim, considerando o cenário atual em que a maioria dos dados apresentam-se em um contexto de *Big Data*, as estratégias de indexação apresentadas são úteis e podem ser adotadas para realizar um processamento eficiente e eficaz dos dados.

Para as funções de similaridade, a *Edit Distance* mostrou-se a mais ineficaz, pois: I) para as bases de dados com atributos que possuem muitos caracteres, o tempo para o cálculo de similaridade é alto; e II) os valores de *F-Measure* não foram significativos, apesar da função possuir a melhor precisão entre as demais. Para as funções *Jaro* e *Jaro Winkler*, os tempos de cálculo da similaridade são equivalentes e, no quesito eficácia, a *Jaro Winkler* apresentou resultados melhores para atributos com poucos caracteres ($BTD10$ e $RC5$). A função *Jaro* apresentou resultados melhores para as outras bases de dados, as quais contém uma quantidade de caracteres maiores ($RP5$, $UBS5$ e $UBS10$).

Para as chaves *Soundex* e *Suffix*, obtiveram-se em ambas resultados significantes, não possuindo uma diferença de *F-Measure* perceptível. Quando utilizadas as duas de maneira combinada, os resultados obtidos não foram melhores e o tempo de execução se mostrou ineficaz. Por fim, como trabalhos futuros pretende-se investigar novas técnicas de BKs, realizar testes estatísticos para fundamentar as conclusões e analisar os casos de falha das técnicas apresentadas.

Referências

Borges, E. N., Galante, R. M., and Gonçalves, M. A. (2008). Uma abordagem efetiva e eficiente para deduplicação de metadados bibliográficos de objetos digitais. In *Proceedings of the 23rd Brazilian Symposium on Databases*, pages 76–90, Campinas, Brasil.

Brizan, D. G. and Tansel, A. U. (2006). A survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, pages 41–50.

Canalle, G. K., Lóscio, B. F., and Salgado, A. C. (2016). Uma estratégia para seleção de atributos relevantes no processo de resolução de entidades. In *Anais do 31º Simpósio Brasileiro de Banco de Dados*, pages 259–264, Salvador, Bahia, Brasil.

Chen, J., Jin, C., Zhang, R., and Zhou, A. (2012). A learning method for entity matching. In *Proceedings of 10th International Workshop on Quality in Databases*, China.

Christen, P. (2012a). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Berlin.

Christen, P. (2012b). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, pages 1537–1555.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, pages 1183–1210.

Hajishirzi, H., Yih, W.-t., and Kolcz, A. (2010). Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 419–426, Geneva, Switzerland.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD*, pages 127–138, New York, USA.

Ioannou, E., Rassadko, N., and Velegrakis, Y. (2013). On generating benchmark data for entity matching. *Journal on Data Semantics*, pages 37–56.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, pages 414–420.

Machado, R. F., Pinheiro, R. F., Machado, K. S., and Borges, E. N. (2016). Contacts deduplication in mobile devices using textual similarity and machine learning. In *Proceedings of the 22nd Brazilian Symposium on Information Systems*, page 22, Florianópolis, Santa Catarina.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, pages 31–88.

Odell, M. and Russell, R. (1918). The soundex coding system. *US Patents*, 1261167.

Vesdapunt, N., Bellare, K., and Dalvi, N. (2014). Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, pages 1071–1082.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, Anaheim, California.