# Análise da Efetividade de Comparadores de Strings para Discriminar Pares Verdadeiros de Pares Falsos no Relacionamento de Registros

**Sergio Miranda Freire[1], Rita de Cássia Braga Gonçalves[2] , André Cipriani Bandarra[2], Miguel GustavoTaranto Villela[2], Alexandre Meire[2], Maria Deolinda Borges Cabral[3], Rosimary Terezinha de Almeida[3]**

[1]Departamento de Tecnologia da Informação e Educação em Saúde, Universidade do Estado do Rio de Janeiro

[2]Agência Nacional de Saúde Suplementar – Rio de Janeiro

[3]Programa de Engenharia Biomédica/COPPE – Universidade Federal do Rio de Janeiro

sergio@lampada.uerj.br, {rita.braga, andre.bandarra, miguel.villela, alexandre.meire}@ans.gov.br, maria.cabral@ibge.com.br, rosal@peb.ufrj.br

***Abstract.*** *The objective of this paper is to analyze the effectiveness of eight string comparators used in record linkage processes. A set of true pairs of names was identified in the databases Hospitalisation Claims and Prepaid Health Plan Users in Brazil. From this set, a set of false pairs was generated and then several string comparators were used in each pair of names. For each comparator, a ROC curve was plotted, its area and the mean time to perform the comparison were calculated. The algorithms presented overall similar performance, but more conclusive results will require a bigger and more representative sample of brazilian names.*

***Resumo.*** *Este artigo visa a analisar a efetividade de oito comparadores de strings utilizados em processos de vinculação de registros. Um conjunto de pares verdadeiros de nomes foi identificado nas bases de dados brasileiras da Comunicação de Internação Hospitalar e do Sistema de Informação do Beneficiário. A partir deles, uma base de pares falsos foi gerada e então os diversos comparadores de strings foram utilizados em cada par de nomes. Para cada comparador, foi construída uma curva ROC, e a sua área e o tempo médio para realizar a comparação foram calculados. Os algoritmos tiveram desempenho global similar, mas resultados mais conclusivos irão exigir uma amostra maior e mais representativa dos nomes brasileiros.*

## 1. Introdução

A vinculação de registros é um campo em crescente expansão com aplicações em muitas áreas da saúde e pesquisa biomédica [Bell and Sethi 2001, Kelman et al 2002]. Ela pode ser vista como a metodologia para identificar em dois ou mais arquivos registros que pertencem à mesma entidade ou encontrar duplicatas dentro de um mesmo arquivo. Chamamos de pares de registros que pertencem à mesma entidade como *pares verdadeiros*, caso contrário esses pares são chamados de *pares falsos*. O termo vinculação de registros originou-se na área de saúde pública, referindo-se a registros de

pacientes que, em geral, são vinculados por meio do nome, data de nascimento, endereço e outras informações disponíveis.

Os diversos métodos usados para a vinculação de registros caem em duas grandes categorias: determinísticos ou probabilísticos. Nos métodos determinísticos, ou são utilizadas regras, ou existem identificadores únicos que permitem determinar quando dois pares de registros se referem à mesma entidade ou não. Nos métodos probabilísticos, são utilizados modelos estatísticos para classificar os pares como pares verdadeiros ou falsos. Métodos probabilísticos podem ser baseados na teoria de vinculação de registros probabilística clássica, como a desenvolvida por Fellegi e Sunter [Fellegi e Sunter 1969], ou em novas abordagens usando entropia máxima e outras técnicas de aprendizagem de máquina [Elfeky et al 2002, McCallum et al 2000].

Todos estes métodos envolvem a comparação dos valores de variáveis, como nomes, endereços, data de nascimento, nome dos genitores, sexo ou outras variáveis associadas às entidades em cada par de registros, para verificar se aquele par corresponde à mesma entidade. Uma entidade pode ser um indivíduo, uma empresa ou qualquer outro tipo de unidade que esteja listada. Destas variáveis, uma das mais importantes para identificar a entidade é o próprio nome. O problema com nomes é que eles não são unívocos (várias entidades podem apresentar o mesmo nome), podem ser apresentados em vários formatos, utilizando abreviaturas ou variações na sua escrita, com erros de digitação, além de poderem mudar ao longo do tempo (casamento, divórcio, etc). Desta forma, algoritmos que comparam *strings* constituem um dos elementos chaves para se determinar se um par de registros representa ou não a mesma entidade.

Diversos comparadores de *strings* tem sido propostos na literatura e diversos estudos foram realizados visando a responder à pergunta sobre qual deles é mais eficiente para ser utilizado no processo de vinculação de registros [Yancey 2005, Cohen et al 2003]. Entretanto, estes algoritmos têm sido avaliados com pares de nomes de outros idiomas, e não o português. Assim fica aberta a questão de que método para comparação de *strings* é mais efetivo para ser utilizado nos processos de vinculação de registros de bases brasileiras.

Assim, o objetivo deste trabalho é comparar a eficiência de algoritmos de comparação de *strings*, utilizando uma base de nomes registrados em bancos de dados do sistema de saúde do Brasil. Este projeto foi aprovado pelo comitê de ética em pesquisa do Hospital Universitário Pedro Ernesto.

## 2. Materiais e Métodos

### 2.1. Fonte de Dados e Amostra de Estudo

Foi utilizada como fonte de dados para estudo a Comunicação de Internação Hospitalar – CIH – no âmbito da saúde suplementar. A CIH constitui-se de um documento padronizado que permite, dentre outras variáveis, a identificação do paciente e a fonte de remuneração da internação. Uma das possíveis categorias da fonte de remuneração é o convênio de plano privado. Nesta categoria, a CIH contém informações sobre a identificação da operadora de plano de saúde, composta por duas variáveis: número do registro na Agência Nacional de Saúde Suplementar – ANS – da operadora de plano privado de assistência à saúde e o código de identificação do beneficiário na operadora.

Em 2005, houve um total de 1.546.845 internações. Do total destas comunicações, 389.133 tinham os campos Registro da Operadora/Código do Beneficiário preenchidos. Estes registros foram pareados com os registros da base do Sistema de Informação de Beneficiário, resultando em 64.746 indivíduos em ambas as bases. Uma inspeção deste conjunto de registros por um dos autores identificou 8.938 pares que são registros do mesmo indivíduo, porém com grafia do nome ligeiramente diferente nas duas bases.

A partir deste ponto, o procedimento para gerar um conjunto de pares de *strings* verdadeiros e pares de *strings* falsos seguiu o procedimento utilizado por Yancey [Yancey, 2005]. Para cada par de nomes da CIH e do SIB pertencentes ao mesmo indivíduo procedeu-se da seguinte forma:

1. Eliminaram-se os acentos, as preposições e os sufixos do nome (Neto, Jr, etc).

2. Para cada nome resultante, separou-se o primeiro elemento (prenome) e o último elemento (sobrenome). Os primeiros elementos de cada nome constituíram um novo par, assim como os últimos elementos de cada nome. Desta forma, temos dois pares de *strings* que pertencem ao mesmo indivíduo.

Como todos os algoritmos irão resultar nos mesmos valores para pares de *strings* que são exatamente iguais, eliminaram-se então todos os pares obtidos acima cujas *strings* eram idênticas. O conjunto de pares resultantes então são os pares verdadeiros, ou seja de *strings* que pertencem à mesma entidade. Como exemplo deste procedimento, consideremos três pares de nomes:

(João Pereira da Silva, Jao P Sillva)

(Maria Costa, Mary Costa

(José  Paranhos, Josue Paranos)

Aplicando o procedimento acima, são gerados os seguintes pares verdadeiros:

(Joao, Jao), (Maria, Mary), (Jose, Josue), (Silva, Sillva), (Paranhos, Paranos)

Para gerar os pares de *strings* falsos, procede-se da seguinte maneira. Para cada par formado pelos primeiros elementos dos nomes, forma-se um conjunto de pares compostos pelo primeiro elemento do par e os segundos elementos de todos os outros pares, com exceção dele próprio. Por exemplo, considerando os conjuntos de pares formados pelos primeiros elementos acima, (Joao, Jao), (Jose, Josue) e (Maria, Mary), são gerados os seguintes pares não verdadeiros (Joao, Josue), (Joao, Mary), (Jose, Jao), (Jose, Mary), (Maria, Jao), e (Maria, Josue). Procedimento similar é realizado para os pares formados pelos segundos elementos dos nomes. No exemplo acima, são então gerados os pares (Silva, Paranos) e (Paranhos, Sillva). O conjunto de pares não verdadeiros é formado por todos os pares gerados acima.

Este procedimento, aplicado aos 8.938 pares identificados acima, resultou em 1.817 pares verdadeiros e 2.175.904 pares falsos.

### 2.2. Algoritmos de Comparação de *Strings*

Diversos comparadores de *strings* tem sido propostos na literatura. Em geral, eles fornecem um valor entre 0 e 1, sendo o valor 1 obtido quando há concordância total entre as *strings*. Um comparador de *string* não é necessariamente uma métrica no sentido matemático e a restrição de seus valores ao intervalo [0,1] é feita principalmente

por conveniência. Neste estudo consideramos os comparadores apresentados por Yancey (2005) e Cohen et al (2003) e para os quais localizamos implementações de fonte aberto disponíveis ou uma descrição do algoritmo que permitisse a sua implementação. Assim os comparadores avaliados foram Jaro, Winkler, Lynch, McClaughlin, Levenshtein, Subsequência Comum mais Longa (SCL), Monge-Elkan, e uma combinação de SCL com Levenshtein (Lev_SCL).

Jaro [Jaro 1995] introduziu um comparador que leva em conta o tamanho das variáveis (número de caracteres) e os tipos de erros que ocorrem comumente com variáveis alfanuméricas. Especificamente para c > 0, o comparador de Jaro é dado por:

$$d_j = \frac{1}{3}\left(\frac{c}{d} + \frac{c}{r} + \frac{c - \tau}{c}\right)$$

onde:

$d_j$: comparador de Jaro;

c: número de caracteres em comum na variável em comparação nos dois arquivos;

d: número de caracteres da primeira string;

r: número de caracteres da segunda string;

$\tau$ : número de transposições de caracteres.

Se c = 0 então $d_j = 0$.

Dois caracteres são considerados comuns se estão em ambas as *strings* sendo que a distância entre as posições do caractere em ambas as *strings* pode ser no máximo a metade do comprimento da menor delas. Transposições são caracteres que estão posicionados em locais diferentes nas *strings* em comparação.

Winkler [Winkler 1994] modificou o comparador de Jaro, levando em conta se os quatro primeiros caracteres são iguais, sendo o valor do comparador dado por:

$$d_w = d_j + i.(1 - d_j)/10$$

onde

$d_w$: comparador de Winkler;

i: quantidade de caracteres iguais nas duas *strings* até a quarta posição;

$d_j$: comparador de Jaro.

McLaughlin [Winkler 1994] leva em conta a similaridade entre os caracteres, sendo seu comparador dado por:

$$d_{mc} = \frac{1}{3}\left(\frac{c + s1 * 0.3}{d} + \frac{c + s2 * 0.3}{r} + \frac{c - \tau}{c}\right)$$

onde:

$d_{mc}$: comparador de McLaughlin;

s1: número de caracteres similares na primeira string;

s2: número de caracteres similares na segunda string;

e as demais variáveis como definidas anteriormente.

Lynch [Winkler 1994] propôs um aumento na nota da comparação quando mais do que a metade dos caracteres, após os quatro primeiros, são iguais. Assim,

$$d_l = d_{mc} + (1 - d_{mc}) * (c - k - 1) / (s1 + s2 + 2 * k + 2)$$

onde:

$d_l$ : comparador de Lynch;

k: número de caracteres iguais até a quarta posição;

e as demais variáveis como definidas anteriormente.

A distância de Levenshtein entre duas *strings* [Stephen 1994] é o número mínimo de passos de edição necessários para transformar uma *string* na outra. As edições permitidas são inserção, exclusão e substituição. Como o método de Levenshtein fornece uma medida de distância, definiu-se uma função de similaridade obtida a partir da transformação da distância calculada para cada par comparado, dada por:

$$Lev = 1 - \frac{Nota\_Lev}{n}$$

onde:

Nota_Lev: nota de similaridade decorrente da distância de Levenshtein;

Lev: distância obtida pelo método de Levenshtein;

n: máximo entre os tamanhos das duas *strings* comparadas.

O comparador Subsequência Comum mais Longa (SCL) [Stephen 1994] utiliza a distância de Levenshtein somente com os passos de inserção e exclusão. O comparador Lev_SCL é a média da distância de Levenshtein e o SCL [Yancey 2005].

O algoritmo de Monge-Elkan [Monge e Elkan 1996] que é uma variante da função de distância de Smith-Waterman com parâmetros de custo particulares, e normalizada para o intervalo [0,1].

As variantes do algoritmo de Jaro (Jaro, Winkler, Lynch e McLaughlin) foram implementados em java a partir do algoritmo em C implementado por Winkler et al [Winkler et al 1994] e testados com a amostra fornecida por Winkler [Winkler 1994]. Para os algoritmos de Levenshtein, SCL e Monge-Elkan foram utilizadas as implementações disponibilizadas pelo NLP Group [NLP Group 2009].

Algoritmos de fonetização criam códigos baseados no modo como palavras ou sílabas são pronunciadas. Por exemplo, o algoritmo Soundex, bastante utilizado em softwares de vinculação de registros, geraria o mesmo valor para “Smith”, “Smithe”. Atualmente, há diversas variantes deste algoritmo. Neste trabalho, os comparadores de *string* apresentados acima também foram avaliados comparando-se os pares de nomes após a transformação dos mesmos por meio de um algoritmo de fonetização da língua portuguesa desenvolvido pela Caixa Econômica Federal [CCS-SIS 1998], e também por meio da variante do algoritmo soundex disponibilizado pelo Apache Commons Project

[Apache Commons Project 2008], com as adaptações para o português propostas pelos autores do software Reclink [Camargo e Coeli 2007]. Para distinguir neste trabalho um algoritmo do outro, o algoritmo de fonetização da Caixa Econômica Federal será denominado *Sompo*.

### 2.3. Medidas de Efetividade dos Algoritmos de Comparação de *Strings*

Os resultados obtidos com a aplicação dos diversos métodos de comparação de *strings* aqui descritos serão avaliados em termos dos resultados apresentados por suas curvas ROC (*Receiver Operating Characteristic*) e a correspondente área sob a curva.

A curva ROC, utilizada em diversas áreas, é um gráfico da sensibilidade x (1 – especificidade) para diferentes valores de corte (limiares). Por sensibilidade (S) entende-se a capacidade de um algoritmo detectar que um par é verdadeiro quando de fato ele é e por especificidade (E) a capacidade de um algoritmo detectar que um par não é verdadeiro quando de fato ele não é. Para um dado limiar c, a sensibilidade é calculada dividindo-se o número de pares verdadeiros para os quais o algoritmo fornece um valor maior que c pelo total de pares verdadeiros. Para o mesmo limiar a especificidade é calculada dividindo-se o número de pares não verdadeiros para os quais o algoritmo fornece um valor menor que c pelo total de pares não verdadeiros. Assim 1 – Especificidade correspondente à proporção de falsos negativos (FN). A área abaixo da curva ROC está associada ao poder discriminante de um algoritmo. Quanto maior for a área, tanto melhor será a capacidade de discriminação do algoritmo. A curva ROC pode servir como orientação para a escolha do melhor ponto de corte para um dado algoritmo, como também para comparar dois (ou mais) algoritmos.

Yancey [Yancey 2005], porém, argumenta que, de toda a extensão da curva ROC, somente a porção compreendida entre 0 e FN, onde FN é o valor que corresponde ao limiar 0,778, ou seja, a porção da curva ROC plotada somente considerando a faixa de limiares entre 0,778 e 1. A razão para isto é que os valores do comparador de *strings* alimenta uma função de interpolação que fornece o peso daquela variável no método de pareamento de registros. Valores do comparador abaixo do limiar acima são sempre associados ao peso mais baixo. Deste modo somente a faixa de limiares entre 0,788 e 1 possui um poder de discriminação. Então para cada comparador, a área sob a porção da curva relevante é dada por:

$$\frac{1}{FN}\int_{0}^{FN} Pr(y \geq t \mid M)\, dPr(y \geq t \mid U)$$

onde:

FN: proporção de falsos negativos;

Pr: Probabilidade;

y: valor do comparador;

t: limiar;

M: conjunto dos pares verdadeiros;

U: conjunto dos pares não verdadeiros.

O tempo médio de execução dos algoritmos foi calculado, medindo-se o tempo gasto para realizar a comparação em cada par de um total de 100.000 pares para cada algoritmo.

## 3. Resultados

A tabela 1 apresenta as áreas sob a curva ROC para cada algoritmo, quando ele compara as *strings* básicas, ou transformadas pelos algoritmos sompo e soundex respectivamente, e o tempo médio para cada algoritmo.

**Tabela 1 – Valores das áreas sob a curva ROC e o tempo médio de execução para cada algoritmo analisado.**

| Algoritmo | Tempo Médio (ms) | Área sob a curva ROC | | |
|---|---|---|---|---|
| | | Básico | Sompo | Soundex |
| **Winkler** | 0.10454 | 0,9976 | 0,9375 | 0,9303 |
| **Lynch** | 0.10386 | 0,9947 | 0,9747 | 0,9677 |
| **Jaro** | 0.10414 | 0,9965 | 0,9062 | 0,8530 |
| **McLaughlin** | 0.10308 | 0,9972 | 0,9563 | 0,9395 |
| **Levenshtein** | 0.06847 | 0,9964 | 0,5974 | 0,7320 |
| **SCL** | 0.12562 | 0,9983 | 0,6094 | 0,7995 |
| **LEV_SCL** | 0.19244 | 0,9976 | 0,7579 | 0,7320 |
| **MongeElkan** | 0.06991 | 0,4670 | 0,5357 | 0,6533 |

Analisando somente a área sob a curva ROC, todos os comparadores tiveram melhor desempenho quando não se altera o nome por meio do sompo ou soundex. Dependendo do algoritmo, o sompo é mais ou menos eficiente do que o soundex. Os algoritmos básicos, com exceção do Monge-Elkan, apresentaram valores similares. A Figura 1 apresenta as curvas ROC para cada algoritmo na região onde os algoritmos passam a apresentar maior discrepância no desempenho (limiares próximos a 1).

É possível observar que, à medida que o limiar aumenta (a curva se aproxima da origem), os algoritmos com melhor desempenho são SCL e Lev_SCL (curvas mais próximas ao vértice superior esquerdo do diagrama.

A figura 2 mostra as curvas ROC para o algoritmo Lev_SCL, Lev_SCL após fonetização e Lev_SCL após o uso do soundex. Observa-se que, à medida que o limiar aproxima-se de 1 o desempenho do soundex se deteriora em relação aos demais. Fato semelhante se observa com os outros algoritmos.

Os tempos médios mais baixos são os de Monge-Elkan e Levenshtein, em torno de 0,06. O mais alto é o de Lev_SCL com 0,19 e os demais em torno de 0,10. Como o algoritmo de Lev_SCL é uma combinação do SCL e do Levenshtein, o tempo médio dele é aproximadamente a soma dos seus componentes.

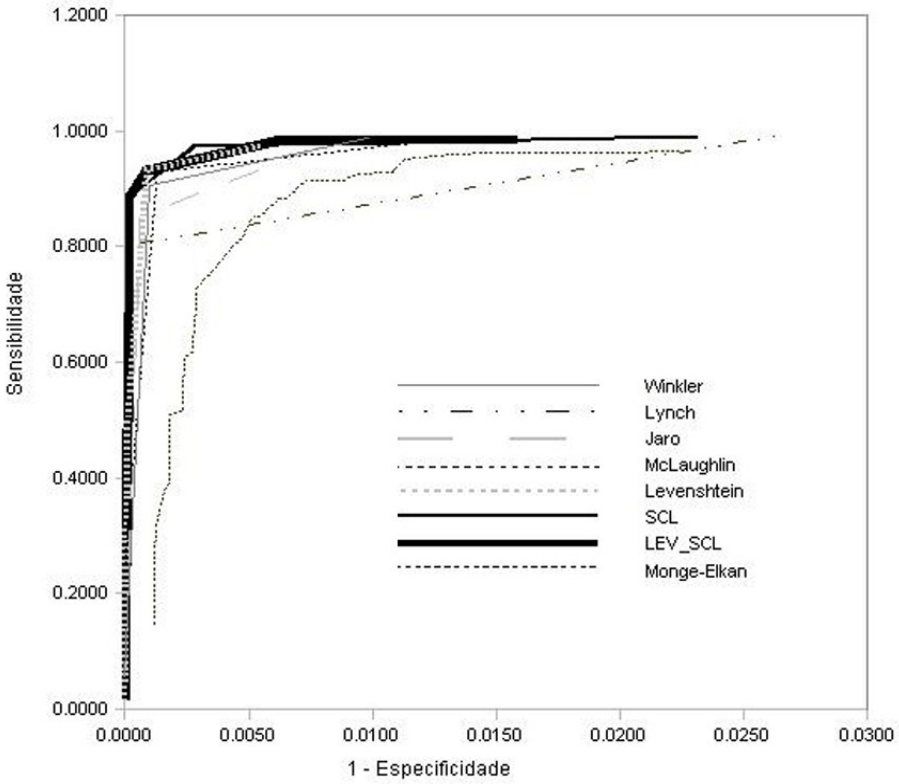


**Figura 1. Curvas ROC dos oito algoritmos na sua versão básica.**

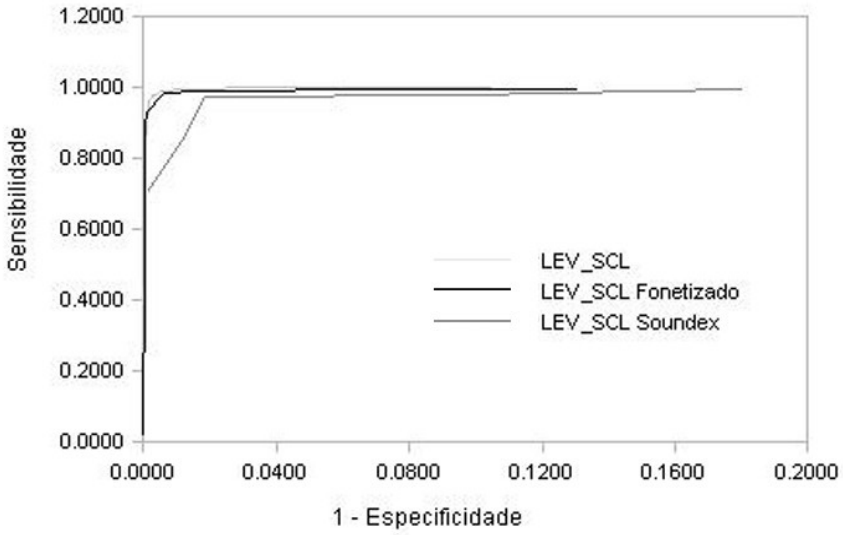


**Figura 2. Curvas ROC para os algoritmos Lev_SCL, Lev_SCL com fonetização, e Lev_SCL com soundex.**

## 4. Discussão

Os resultados deste estudo não diferem marcadamente daqueles obtidos por Yancey, embora este autor não tenha incluído em seu estudo o algoritmo de Monge-Elkan. Os valores obtidos neste estudo para o complemento da especificidade são

bastante mais elevados do que os de Yancey. Uma explicação para esta discrepância pode estar na amostra bastante pequena deste estudo.

O número elevado de pares falsos é esperado, dada a natureza combinatória de geração dos mesmos a partir dos pares verdadeiros. Isto simula o que acontece frequentemente na prática, onde o número de pares verdadeiros é muito menor do que o conjunto de pares possíveis. Porém, o número de pares verdadeiros é bastante pequeno em nossa amostra, particularmente se compararmos com o número de pares verdadeiros da amostra utilizada por Yancey [Yancey 2005] em seu trabalho, pouco mais de 120.000 pares verdadeiros. Uma consequência disto é que o número de falsos negativos é bastante reduzido, mesmo para limiares próximos a 1, o que dificulta a comparação de algoritmos na região da curva ROC para limiares próximos a 1. No Brasil, ainda não dispomos de uma base de referência para realizar estudos sobre vinculação de registro. A comparação das bases do SIB e CIH, utilizando o identificador do beneficiário, não produziu um número grande de vínculos verdadeiros.

Apesar de os tempos para a comparação das *strings* serem diferentes, sendo o mais veloz três vezes mais rápido do que o mais lento, este fator pode não ser relevante para a determinação final do tempo gasto para a realização de uma vinculação de registros, já que o tempo de acesso à base de dados provavelmente será o determinante mais crítico.

Neste trabalho, utilizou-se um algoritmo de fonetização para nomes brasileiros fornecidos pela Caixa Econômica Federal. Entretanto, não é do conhecimentos dos autores que uma avaliação rigorosa deste algoritmo tenha sido realizada.

O software nacional, denominado Reclink, bastante utilizado no Brasil para realizar vinculação de registros na área de saúde, utiliza o algoritmo de Levenshtein para realizar a comparação de registros e o algoritmo soundex para blocagem [Camargo e Coeli 2000 e 2007]. Este estudo sugere que o algoritmo de Levenshtein pode não ser o mais eficiente para realizar comparação de *strings* e que o soundex pode não ser a melhor opção para blocagem. Como continuação deste trabalho, pretende-se obter uma amostra mais representativa da população de nomes no Brasil de modo a se obter resultados mais conclusivos. Também deve ser levado em consideração que o resultado de um processo de vinculação de registros depende de um conjunto de etapas e não somente da comparação de *strings*.

## 5. Agradecimentos

Os autores agradecem o apoio financeiro recebido pelo CNPq, Edital MCT- CNPq/ANS – Nº 25/2007, e as sugestões dos revisores anônimos para a melhoria do texto.

## Referências

Apache Commons Project (2008), “Implementations of common encoders and decoders”. http://commons.apache.org/codec. Último acesso em 10/03/2009.

Bell, G.B. and Sethi, A. (2001), “Matching Records in a National Medical Patient Index”, *Communications of the ACM*, 44(9):83–88.

Camargo, K.R., Coeli, C.M. (2000), “Reclink: aplicativo para o relacionamento de bases de dados, implementando o método probabilistic record linkage”, *Cad. Saúde Pública*, 16(2):439-447.

Camargo, K.R., Coeli, C.M. (2007), “Reclink III: Relacionamento Probabilístico de Registros, Versão 3.1.6.3160. Manual, Rio de Janeiro.

CCS-SIS (1998), “Tfonetizar. Consórcio de Componentes de Software para Sistemas de Informação em Saúde”. http:www.datasus.gov.br/ccssis/tfonetiz.htm. Último acesso em 10/03/2009.

Cohen, W.W., Ravikumar, P. and Fienberg, S.E. (2003), “A Comparison of String Distance Metrics for Name-Matching Tasks”. http://secondstring.sourceforge.net/doc/iiweb03.pdf. Último acesso em 11/01/2009.

Elfeky, M.G., Verykios, V.S. and Elmagarmid, A.K. (2002), “TAILOR: A Record Linkage Toolbox”. In Proc. of the 18th Int. Conf. on Data Engineering. IEEE, 2002.

Fellegi, L.and Sunter, A (1969), “A Theory for Record Linkage”. *Journal of the American Statistical Society*, 64:1183–1210.

Jaro, M. A. (1978), "UNIMATCH: A Record Linkage System, User's Manual," Washington, DC: U.S. Bureau of the Census.

Jaro, M. A. 1995. Probabilistic linkage of large public health data files (disc: P687-689). *Statistics in Medicine* 14:491–498.

Kelman, C.W., Bass, A.J. and Holman, C.D. (2002), “Research use of linked health data - a best practice protocol”. *Aust N Z J Public Health*, 26:251–255.

McCallum, A., Nigam, K. and Ungar, L. (2000), “Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integation”. In Proc. of the Sixth Int. Conf. on KDD, p. 169–170.

Monge, A. and Elkan, C. (1996). The field-matching problem: algorithm and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

NLP Group (2009), “SimMetrics, an open source extensible library of Similarity or Distance Metrics”. http://www.dcs.shef.ac.uk/~sam/simmetrics.html. Último acesso em 10/03/2009.

Stephen, G. A. (1994), String Searching Algorithms. World Scientific Publishing Co. Pte. Ltd.

Winkler, W. E. (1994), “Advanced Methods for Record Linkage”, US Census Bureau Report RR94/05. http://www.census.gov/srd/www/byname.html. Último acesso em 10/03/2009.

Winkler, W.E., McLaughlin, G., Jaro, M. And Lynch, M. (1994), “strcmp95.c Version 2”. http://www.census.gov/geo/msb/stand/strcmp.c. Último acesso em 10/01/2009.

Yancey. W. (2005), “Evaluating String Comparator Performance for Record Linkage”, Report *RRS2005/05*, US Bureau of the Census.: http://www.census.gov/srd/www/byname.html. Último acesso em 10/01/2009.