Deduplicação de Contatos em Dispositivos Móveis Utilizando Similaridade Textual e Aprendizado de Máquina

Alternative title: Contacts Deduplication in Mobile Devices Using Textual Similarity and Machine Learning

Rafael F. Machado
Centro de Ciências Computacionais
Univ. Federal do Rio Grande
Av. Itália km 8, Rio Grande,RS
rafaelmachado@furg.br

Rafael F. Pinheiro
Centro de Ciências Computacionais
Univ. Federal do Rio Grande
Av. Itália km 8, Rio Grande,RS
rafaelpinheiro@furg.br

Karina S. Machado
Centro de Ciências Computacionais
Univ. Federal do Rio Grande
Av. Itália km 8, Rio Grande,RS
karinamachado@furg.br

Eduardo N. Borges
Centro de Ciências Computacionais
Univ. Federal do Rio Grande
Av. Itália km 8, Rio Grande,RS
eduardoborges@furg.br

RESUMO

Informações redundantes e muitas vezes incompletas reduzem consideravelmente a produtividade oferecida pelos dispositivos móveis. Este artigo apresenta um método que identifica contatos duplicados, ou seja, registros que representam a mesma pessoa ou organização, coletados automaticamente de múltiplas fontes de dados. Os contatos são comparados utilizando diversas funções de similaridade cujos escores são combinados por um modelo de classificação baseado em árvores de decisão, que elimina a necessidade do especialista para configurar limiares de similaridade. Os experimentos realizados mostram que o método proposto identificou corretamente até 92% dos contatos duplicados.

Palavras-Chave

Deduplicação, gerência de contatos, similaridade, aprendizado de máquina.

ABSTRACT

This paper presents a method that identifies duplicate contacts, i.e., records representing the same person or organization, automatically collected from multiple data sources. Contacts are compared using several similarity functions, of which scores are combined by a classification model based on decision trees, which eliminates the need for an expert to manually configure similarity thresholds. The experiments show that the proposed method correctly identified up to 92% of duplicate contacts.

SBSI 2016, May 17th-20th, 2016, Florianópolis, Santa Catarina, Brazil

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms, Experimentation

Keywords

Deduplication, contacts management, similarity, machine learning.

1. INTRODUÇÃO

Nos últimos anos, a Internet e a Web revolucionaram o modo como as pessoas se comunicam. Com a explosão do número de aplicações Web disponíveis, os usuários tendem a acumular diversas contas em diferentes serviços como *e-mail*, redes sociais, *streams* de música e vídeo, lojas virtuais, entre outros. O avanço da tecnologia e a redução do seu custo têm proporcionado o acesso a todos os serviços mencionados de qualquer lugar, a partir de dispositivos móveis como *smartphones* e *tablets*.

Gerenciar informações provenientes de múltiplos serviços ou aplicações é uma tarefa complexa para o usuário. Alguns serviços básicos do dispositivo móvel podem ser prejudicados pela redundância da informação coletada automaticamente por diferentes aplicações. Por exemplo, navegar na lista de contatos com tantas informações repetidas e muitas vezes incompletas reduz consideravelmente a produtividade que o dispositivo móvel pode oferecer.

A Figura 1 apresenta uma porção de uma lista de contatos real composta por dez registros, obtidos de fontes de dados

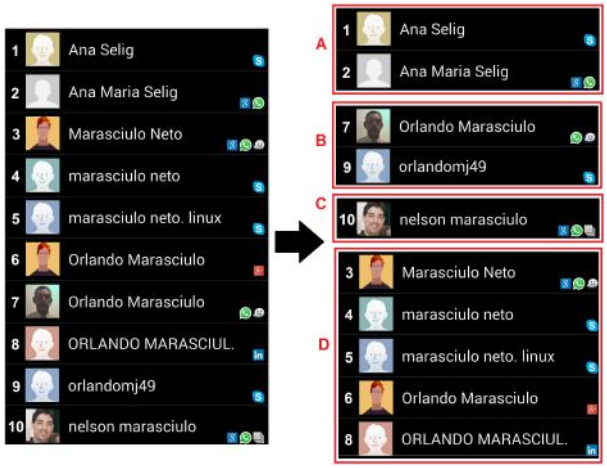

Figura 1: Exemplo de lista de contatos (à esquerda) incluindo registros duplicados e o resultado esperado da deduplicação (à direita).

distintas representadas pelos ícones à direita. Algumas informações já estão combinadas de algumas fontes, como é o caso do registro 3. Entretanto, os registros 4, 5, 6 e 8 representam a mesma pessoa e poderiam ser integrados ao registro 3 formando o conjunto D. Ainda poderiam ser integrados os pares de registros $A = \{1, 2\}$ e $B = \{7, 9\}$ (pai de A), pois representam o mesmo contato. O registro 10 não apresenta duplicatas, portanto permanece isolado.

Sistemas operacionais populares para dispositivos móveis, como iOS [13] e Android [1], oferecem de forma nativa uma funcionalidade de associação de contatos em que o usuário precisa selecionar os registros que deseja combinar. Esta tarefa de associação, além de custosa, é armazenada no dispositivo. Se por ventura o usuário perder o dispositivo ou tiver que reinstalar o sistema, os contatos restaurados do backup de sua conta online não estarão associados.

Este trabalho estende o método de deduplicação publicado em [15], que utiliza múltiplas funções de similaridade para comparar os campos que descrevem os contatos coletados de múltiplas fontes. Neste trabalho, os escores retornados pelas funções são utilizados para treinar um modelo de classificação, baseado em árvores de decisão, podendo ser utilizado como parte da estratégia de associação (integração) automática de contatos. O novo método é comparado às estratégias implementadas por um conjunto de aplicativos para gerência de contatos disponíveis gratuitamente. A qualidade do método ainda é avaliada de forma experimental sobre uma base de dados real com quase 2000 registros.

O restante do texto está organizado da seguinte forma. A Seção 2 apresenta um estudo sobre um conjunto de cinco aplicativos para gerência de contatos. Na Seção 3 são revisados trabalhos da literatura científica sobre conceitos fundamentais para o entendimento do trabalho proposto. A Seção 4 especifica o método proposto para deduplicação de contatos. O protótipo desenvolvido e os resultados da avaliação experimental são discutidos na Seção 5. Por fim, na Seção 6, são apresentadas as conclusões e apontados alguns trabalhos futuros.

2. *APPS* PARA GERÊNCIA DE CONTATOS

As lojas online Google Play[1], Apple App[2] e Windows Phone Store[3] disponibilizam uma série de aplicativos para gerência de contatos, entretanto a grande maioria tem como objetivo facilitar a inserção, edição, organização e compartilhamento de informações sobre contatos de forma mais intuitiva para o usuário do que usando os aplicativos instalados por padrão nos sistemas operacionais Android, iOS e Windows Phone. Poucos aplicativos focam no problema da identificação e eliminação de contatos duplicados.

Uma solução completa de integração de dados deve estabelecer métodos específicos que solucionem as seguintes tarefas: importar os registros de dados de diferentes fontes heterogêneas; transformar os dados de forma a obterem uma representação comum, ou seja, um esquema compatível; identificar aqueles registros semanticamente equivalentes, representando o mesmo objeto; mesclar as informações provenientes das múltiplas fontes; apresentar ao usuário final o conjunto de registros sem informação duplicada [14]. Os aplicativos estudados concentram-se apenas nas três últimas tarefas porque utilizam métodos disponíveis na API do sistema operacional para ler os registros em um mesmo esquema.

O aplicativo Limpador de Contatos [19] remove contatos duplicados da agenda comparando apenas os números de telefone. A interface gráfica, apresentada na Figura 2 (à esquerda), possui um único botão que quando acionado remove as duplicatas sem qualquer interação do usuário. É exibida uma notificação com o número de contatos excluídos. Não é possível visualizar os contatos detectados como réplicas e tampouco restaurar a agenda original.

Já Duplicate Contacts [2] permite visualizar os contatos duplicados e selecionar os registros a serem excluídos. Uma seleção prévia é apresentada automaticamente para o usuário usando a igualdade dos números de telefone (vide Figura 2). Também é possível configurar um arquivo de backup com a agenda no estado anterior às modificações.

O terceiro *app* estudado, denominado Duplicate Contacts Delete [8], tem as mesmas funcionalidades dos apresentados anteriormente, mas utiliza, além dos números de telefone, o nome do contato para identificar duplicatas. A interface gráfica com destaque para os botões na parte inferior é apresentada na Figura 2.

Os três aplicativos apresentados, além de serem implementados exclusivamente para o Android, permitem apenas eliminar contatos duplicados. Também foram analisados outros dois aplicativos que oferecem funções de integração das informações redundantes e associação dos contatos.

Contact Merger [17], disponível para Android e Windows Phone, combina todos os números de telefone de contatos com o mesmo nome, mas mantém apenas um dos nomes para contatos com o mesmo telefone. Esta segunda estratégia é perigosa porque informação relevante pode ser perdida no processo de integração. Por exemplo, o nome de um contato pode ser substituído por um apelido, como no caso de integrar "*Dado*" e "*Eduardo Oliveira*". Também é possível que um qualificador de local de trabalho importante na descrição do contato seja removido, como na associação de "*Renata Santander*" e "*Renata*". A interface deste aplica-

[1] `http://play.google.com`
[2] `http://www.apple.com/appstore`
[3] `http://www.windowsphone.com/store`

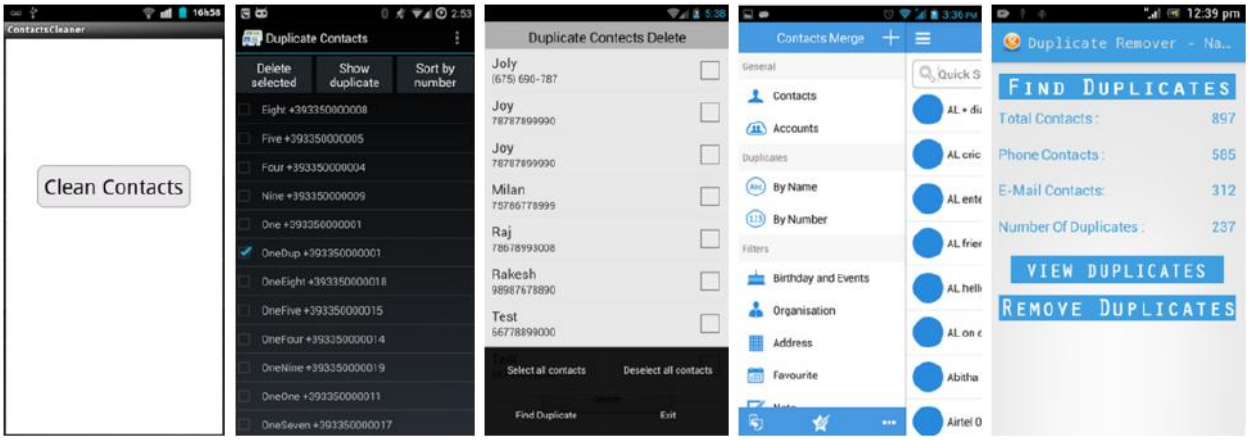

Figura 2: Interface gráfica dos aplicativos estudados (da esquerda para a direita): Limpador de Contatos, Duplicate Contacts, Duplicate Contacts Delete, Contact Merger e Duplicate Contacts Manager.

tivo é bastante atraente (vide Figura 2) e permite o acesso a outras funcionalidades na gerência de contatos, tais como aniversários, endereços e contatos favoritos.

Por fim, Duplicate Contacts Manager [20] se destaca porque também utiliza o *e-mail* na deduplicação. Após detectar os registros duplicados, são exibidas estatísticas sobre cada tipo de contato e o número de duplicatas encontradas, que podem ser visualizadas ou removidas diretamente. A Figura 2 (à direita) apresenta a interface gráfica destacando a funcionalidade mencionada. Infelizmente, a integração está disponível apenas na versão paga e não pode ser testada. Este aplicativo está disponível apenas para o Android.

A Tabela 1 resume as propriedades dos aplicativos estudados e as compara com as características do método proposto neste artigo. Para cada aplicativo são apresentados os campos utilizados na deduplicação, a função de comparação desses campos, o tipo de alteração (exclusão ou integração de contatos duplicados) e a possibilidade de restauração da agenda original.

O diferencial e as principais contribuições do trabalho proposto são: o uso de funções de similaridade textual para comparação dos nomes e *e-mails*, permitindo que muitos dos casos apresentados no exemplo motivacional da Figura 1 sejam identificados como o mesmo contato; e o uso de um modelo de classificação que combina os escores retornados pelas funções de similaridade automaticamente, eliminando a necessidade do especialista para configuração de um limiar de similaridade para cada função. Como o foco do trabalho é o método de identificação de registros de contatos duplicados, a integração e o backup não são aplicáveis. Uma solução para associação dos contatos é um dos trabalhos futuros citados na Seção 6.

3. FUNDAMENTAÇÃO TEÓRICA

A tarefa de identificar registros duplicados que se referem a mesma entidade do mundo real é denominada deduplicação [4]. Nos últimos anos, diversos métodos foram propostos para a deduplicação de registros, principalmente no contexto da integração de dados relacionais [9, 6, 11, 10, 5, 3]. Não foram encontradas na literatura abordagens específicas para deduplicar registros de contatos em dispositivos móveis.

Grande parte dos métodos propostos para identificação de duplicatas utiliza o conceito de medida de similaridade textual, calculada através de uma função de similaridade ou de distância. As subseções seguintes apresentam as funções utilizadas no método proposto [7]. Elas foram escolhidas porque cobrem diferentes estratégias de comparação e são adequadas para identificar similaridades em nomes próprios. Todas as funções retornam escores no intervalo $[0, 1]$.

3.1 *Jaccard*

Sejam A e B cadeias de caracteres representadas por conjuntos de palavras. A função *Jaccard* calcula a similaridade entre A e B de acordo com a Equação 1, ou seja, retorna a razão entre a quantidade de palavras compartilhadas pelas cadeias de caracteres e todas as palavras que as compõem.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

3.2 *Levenshtein*

Sejam a e b cadeias de caracteres, a distância de *Levenshtein* ($dist$) resulta no menor número de inserções, exclusões ou substituições de caracteres necessárias para transformar a em b. A similaridade é calculada como o complemento da distância normalizada, conforme a Equação 2, em que $\max(a, b)$ é o número de caracteres da maior cadeia.

$$Levenshtein(a, b) = 1 - \frac{dist(a, b)}{\max(a, b)} \quad (2)$$

3.3 *JaroWinkler*

Seja m o número de correlações entres os caracteres e t o número de transposições, a função *Jaro* calcula a similaridade entre as cadeias de caracteres a e b de acordo com a Equação 3. *JaroWinkler* é uma variação de *Jaro* que pondera prefixos, de tamanho p, presentes nas duas cadeias. Esta função é definida pela Equação 4.

$$Jaro(a, b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m - t}{m}\right) \quad (3)$$

$$JaroWinkler(a, b) = Jaro(a, b) + \frac{p}{10}(1 - Jaro(a, b)) \quad (4)$$

Tabela 1: Características dos aplicativos estudados e do trabalho proposto.

Aplicativo	Campos	Comparação	Aprendizagem
Limpador de Contatos	telefone	igualdade	não
Duplicate Contacts	telefone	igualdade	não
Duplicate Contacts Delete	telefone, nome	igualdade	não
Contact Merger	telefone, nome	igualdade	não
Duplicate Contacts Manager	telefone, nome, *e-mail*	igualdade	não
Trabalho proposto	telefone, nome, *e-mail*	**similaridade**	**sim**

3.4 *MongeElkan*

Seja $A = \{a_1, ..., a_K\}$ e $B = \{b_1, ..., b_L\}$ cadeias de caracteres representadas por conjuntos de K e L palavras respectivamente. A função *MongeElkan* executa para cada par de palavras uma função de similaridade auxiliar, geralmente *Levenshtein*, retornando a média das máximas similaridades conforme a Equação 5.

$$MongeElkan(A, B) = \frac{1}{K} \sum_{i=1}^{K} \max_{j=1}^{L} sim(a_i, b_j) \qquad (5)$$

4. DEDUPLICAÇÃO DE CONTATOS

A deduplicação pode ser uma tarefa bastante difícil, devido principalmente aos problemas: uso de acrônimos, diferentes estilos de formatação, estrutura dos metadados distinta, variação na representação do conteúdo, omissão de determinados campos e omissão de conteúdo relevante. Na deduplicação de contatos em dispositivos móveis não é comum o uso de acrônimos e os dados não possuem um determinado estilo. A estrutura dos registros é a mesma, porque as API dos sistemas operacionais permitem recuperar todos os registros no mesmo formato, mesmo que tenham sido coletados automaticamente de diferentes redes sociais ou outras aplicações.

Portanto, o foco da deduplicação de contatos é resolver o problema da variação e omissão de conteúdo, que é muito frequente e ainda mais grave do que em outros contextos como em bibliotecas digitais. Enquanto muitos contatos duplicados compartilham apenas o primeiro nome, referências bibliográficas apresentam diferentes representações dos autores (ordem dos nomes e abreviações) e pouca variação no título das publicações. Além disso, contam com outros metadados relevantes, como o ano e o veículo de publicação. Já a grande maioria dos contatos contam apenas com uma informação adicional além do nome: número(s) de telefone e identificador único na aplicação da qual foi coletado.

O método proposto é dividido em quatro fases principais: coleta e preprocessamento, cálculo das similaridades, classificação de pares duplicados e agrupamento de contatos equivalentes (vide Figura 3).

4.1 Coleta e Preprocessamento

Na primeira fase são coletados os contatos do dispositivo provenientes da memória interna, cartões SIM e de contas vinculadas a outras aplicações locais ou na nuvem, como mensageiros e redes sociais. Para cada contato importado, são selecionados e armazenados registros que contêm campos que representem nome, telefone ou *e-mail*.

Os nomes são preprocessados removendo-se acentuação, caixa alta e caracteres diferentes de letras ou números. É armazenado em um novo campo o login do *e-mail* (sem o domínio). Por fim, são mantidos apenas os 10 algarismos finais do número de telefone, de modo a incluir o código DDD seguido dos oito dígitos do telefone. A Seção 6 relata a inclusão do nono dígito como trabalho futuro.

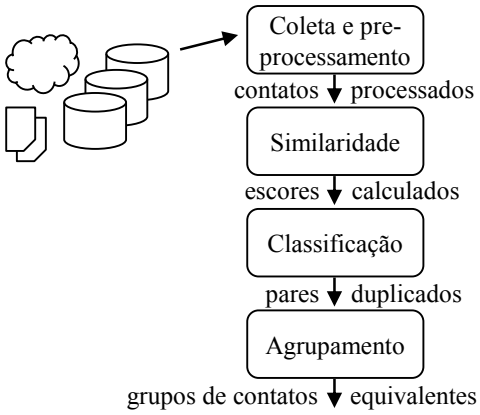

Figura 3: Fases do método proposto para identificação de contatos duplicados.

4.2 Similaridade

Na segunda fase os registros são combinados em pares. Aqueles que compartilham pelo menos um número de telefone ou endereço de *e-mail* (casamento por igualdade) são diretamente identificados como pares duplicados e enviados para a fase *Agrupamento*.

Sobre os demais registros são aplicadas as seguintes funções de similaridade sobre seus campos:

- Levenshtein (logins);
- Jaccard (nomes);
- JaroWinkler (nomes);
- MongeElkan (nomes).

A Tabela 2 exemplifica um par de contatos e os escores retornados pelas funções.

4.3 Classificação

Na terceira fase, os escores retornados pelas funções de similaridade compõem novos registros, junto de um atributo que representa a duplicidade do par. Estes registros são utilizados como entrada de um modelo de classificação que rotula cada par de contatos como duplicados ou distintos. Este modelo é uma árvore de decisão treinada pelo algoritmo C4.5 [18].

Os pares rotulados como duplicados nesta fase são adicionados aos pares identificados anteriormente (telefones ou *e-mails* iguais).

Tabela 2: Par de contatos e os escores retornados pelas funções de similaridade.

Nome	Login	*Levenshtein*	*Jaccard*	*JaroWinkler*	*MongeElkan*
Mateus Gabriel Muller Mateus Muller	mateusmuller mateusmuller2	0,92	0,66	0,6	0,75

4.4 Agrupamento

Por fim, na quarta e última fase, os contatos equivalentes podem ser agrupados utilizando duas estratégias diferentes: (i) cada registro é similar a pelo menos um registro do mesmo grupo e (ii) todos os registros de um grupo são similares entre si.

Para implementar estas estratégias é definido um grafo de duplicatas em que cada vértice representa um contato e as arestas representam a duplicidade. Sobre este grafo são executados dois algoritmos [12]:

- *Single Link* – que retorna um grupo para cada componente conexa do grafo, implementando a primeira estratégia;
- *Click* – que retorna grupos representando subgrafos completos, implementando a segunda estratégia.

A Figura 4 ilustra o resultado dos algoritmos de agrupamento considerando o grafo de duplicatas à esquerda como entrada.

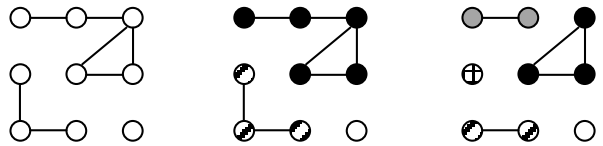

Figura 4: Exemplo de agrupamento *Single Link* (centro) e *Click* (à direita).

5. AVALIAÇÃO EXPERIMENTAL

Esta seção descreve os experimentos conduzidos com o objetivo de validar o método de deduplicação de contatos proposto neste artigo. A Figura 5 mostra a interface do protótipo implementado. Ele foi programado na linguagem Java utilizando o SDK do Android. São exibidas a tela inicial, o menu de funções, a lista de contatos de um dispositivo e o resultado do agrupamento dos contatos duplicados utilizando os algoritmos *Single Link* e *Click*.

Foi utilizada uma base de dados real, disponibilizada por um voluntário, com exatos 1962 contatos importados de múltiplas fontes de dados: memória interna, cartão SIM, Skype, Facebook, LinkedIn, GMail e Google+. A qualidade do método foi avaliada utilizando o número de instâncias corretamente classificadas e a revocação [16] para a classe *contatos duplicados*. Foi utilizada a ferramenta Weka[4] [21] para treinamento e avaliação dos modelos de classificação.

Foram selecionados todos os contatos que continham pelo menos um nome, além de um número de telefone ou *e-mail*. Depois do preprocessamento restaram 1072 contatos válidos. Estes foram combinados dois a dois. Foram excluídos pares de contatos com telefones ou *e-mails* iguais (duplicatas óbvias detectadas com casamento por igualdade), totalizando 574.044 pares, dentre os quais apenas 66 representam contatos duplicados. Para cada par foram executadas as funções de similaridade apresentadas. Aos escores retornados foram adicionados os atributos relativos à duplicidade e compostos os registros correspondentes.

Como as classes são muito desbalanceadas (66 pares duplicados e 573.978 distintos), não foi possível adotar uma estratégia de avaliação de modelos usual como a validação cruzada. Portanto, foram gerados cinco conjuntos de treinamento com 1000 instâncias cada. Estas instâncias foram selecionadas aleatoriamente, distribuídas da seguinte maneira: 33 representando contatos duplicados (metade dos 66 disponíveis) e 967, distintos. Para cada conjunto de treinamento foi gerado um conjunto de teste correspondente com o complemento das instâncias disponíveis, ou seja, contendo os 573.044 registros restantes, dentre os quais estão presentes os outros 33 pares de contatos duplicados.

A Figura 6 apresenta a distribuição de valores de cada atributo (escores de similaridade) em função da classe para um dos cinco conjuntos de dados gerados. A cor vermelho escuro representa os contatos duplicados.

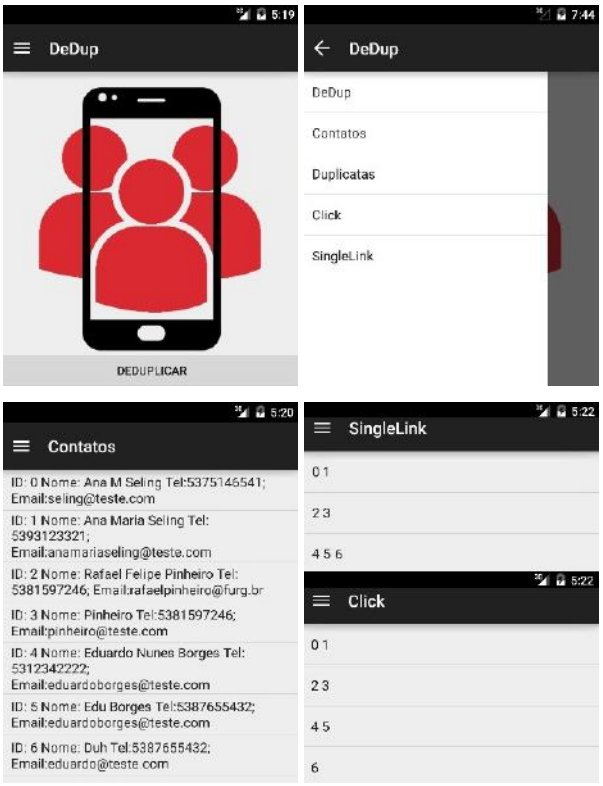

Figura 5: Protótipo implementado para o Android, destacando o resultado da deduplicação.

[4] `http://www.cs.waikato.ac.nz/ml/weka`

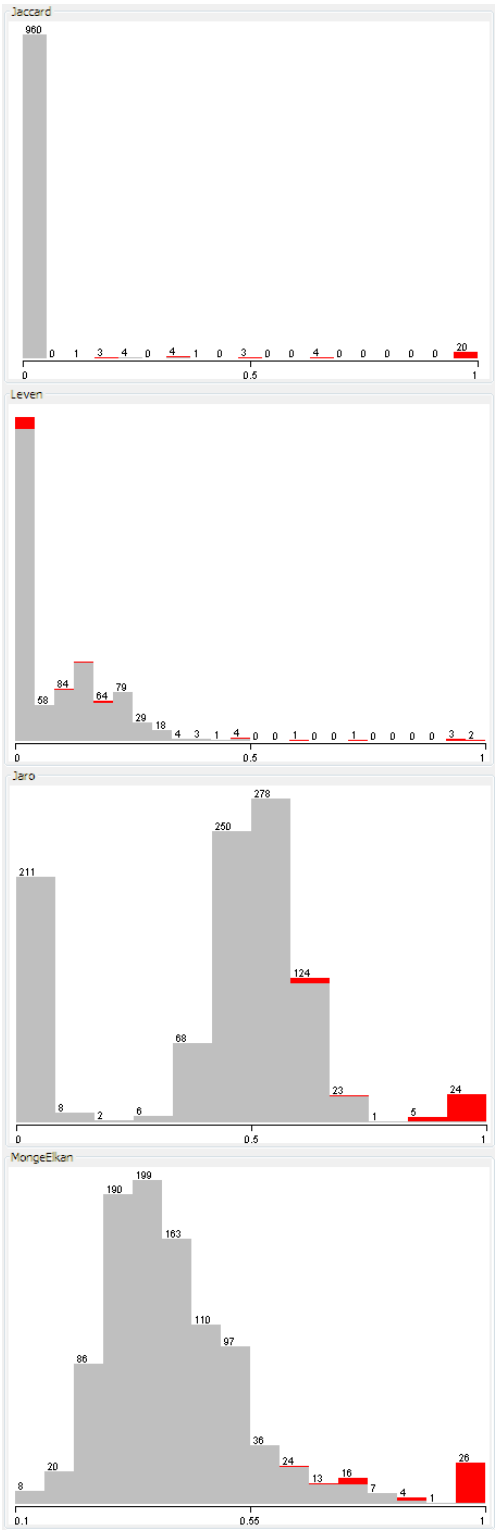

Figura 6: Distribuição dos escores de similaridade em função da classe para um dos conjuntos de treinamento.

Nenhuma das funções de similaridade isoladamente é capaz de separar corretamente a grande maioria dos contatos duplicados. Além disso, um usuário especialista teria muita dificuldade em atribuir um limiar para cada função e/ou definir uma forma de combinar os escores adequadamente. Os resultados apresentados a seguir evidenciam a contribuição do uso de aprendizado de máquina no processo de deduplicação de contatos.

5.1 Resultados

A Tabela 3 resume os resultados obtidos pelos modelos treinados, ou seja, a quantidade de pares de contatos duplicados ou distintos classificados correta e incorretamente. Para cada conjunto de dados (D) é exibido: a frequência de verdadeiros positivos (VP), a frequência de falsos negativos (FN), a frequência de verdadeiros negativos (VN), a frequência de falsos positivos (FP), a porcentagem do total de instâncias classificadas corretamente (ICC) e a revocação da classe correspondente aos contatos duplicados (R_{dup}), ou seja, a porcentagem de pares duplicados identificados corretamente. As últimas linhas mostram a média e o desvio padrão dos valores.

Tabela 3: Resultado da deduplicação utilizando o classificador C4.5.

D	VP	FN	VN	FP	$ICC(\%)$	$R_{dup}(\%)$
1	32	1	572309	702	99,9	97,0
2	28	5	572308	703	99,9	84,8
3	27	6	571452	1559	99,7	81,8
4	32	1	570835	2176	99,6	97,0
5	33	0	570085	2926	99,5	100
M	30,4	2,6	571398	1613	99,7	92,1
DP	2,7	2,7	962	962	0,2	8,2

A qualidade geral dos modelos pode ser observada pela alta taxa de instâncias de teste classificadas corretamente. Em média, 97% dos pares de contatos foram identificados na classe certa. Quando observada apenas a classe *duplicados*, foram identificados de 27 a 33 pares, resultando na média de 30,4 (92,1%) pares de contatos duplicados. O número de falsos negativos foi significativo apenas nos conjuntos de dados 2 e 3, fazendo com que a revocação caísse para 84,8 e 81,8% respectivamente.

Apesar dos bons resultados apresentados, muitos pares distintos foram incorretamente classificados como duplicados (1613 em média). O alto desvio padrão ocorreu porque os 967 registros aleatoriamente selecionados para compor os conjuntos de treinamento não são suficientemente representativos. Entretanto, o objetivo deste trabalho é classificar corretamente os casos em que os contatos são duplicados. A respeito dos falsos negativos, em média, apenas 2,6 dos 33 (7,9%) dos contatos duplicados foram classificados incorretamente.

A Figura 7 mostra o modelo aprendido pelo algoritmo C4.5 para o conjunto de treinamento 1. A classe 1 correspondente a *contatos duplicados* e 0 a *contatos distintos*. A raiz da árvore de decisão apresenta o atributo mais discriminatório, indicando a importância da similaridade entre os nomes dos contatos segundo a função *Jaccard*. Para escores menores ou iguais a 0,12, o modelo foi capaz de classificar 961 dos 967 (99,4%) pares distintos disponíveis no conjunto de treinamento. Para escores maiores que 0,33, foram classificados 28 dos 33 (84,8%) contatos duplicados. Os casos

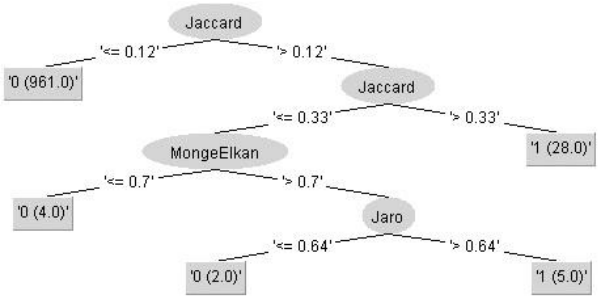

Figura 7: Modelo gerado a partir do conjunto de treinamento 1.

que não foram identificados pelo dois primeiros nós da árvore foram classificados pelos nós folhas que correspondem aos escores retornados pelas funções de similaridade *MongeElkan* e *JaroWinkler*.

A menor revocação R_{dup} = 81,8% apresentada no conjunto de teste 3 pode ser explicada pela simplicidade do modelo gerado pelo treinamento correspondente (Figura 8). A árvore de decisão contém apenas um atributo que representa os escores retornados pela função *JaroWinkler*. Para escores menores ou iguais a 0,84, todos os 967 pares distintos disponíveis no conjunto de treinamento são identificados corretamente, mas ocorrem 2 erros de classificação dos pares duplicados. Valores maiores que 0,84 identificaram corretamente 31 dos 33 (93,9%) pares duplicados.

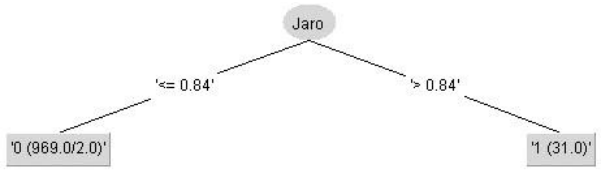

Figura 8: Modelo gerado a partir do conjunto de treinamento 3.

5.2 Análise dos casos de falha

A Tabela 4 apresenta os casos em que o método proposto falhou ao identificar os pares de contatos duplicados, ou seja, os falsos negativos. Para cada tipo de falha e conjunto de dados ($D_i|1 \leq i \leq 5$) é exibido o número de casos. As falhas são erros de classificação causados por: uma função de similaridade não usada no modelo, representada pela fonte tachada, ou um escore retornado por determinada função de similaridade menor (⇓) ou maior (⇑) do que o limiar da regra que classifica a instância avaliada. A seguir são apresentado alguns exemplos de registros e os motivos pelos quais encaixam-se em determinado tipo de falha.

Em D_1, o par de contatos cujos nomes são "*Leonardo C3*" e "*Leonardo Emmendorfer*" não são identificados porque *MongeElkan* retorna 0,5. Este escore é menor que 0,7 , portanto foi classificado por um nó folha de classe *contatos distintos* (vide Figura 7). Este caso de falha é muito difícil de solucionar porque *C3* é um qualificador que se refere ao local de trabalho do contato, enquanto *Emmendorfer* é o sobrenome.

Já em D_3, os contatos "*Nyland*" e "*Nathan Nyland*", apesar de retornaram escore máximo de *MongeElkan*, atingem

Tabela 4: Casos de falha na identificação de contatos duplicados (falsos negativos).

Falha	D_1	D_2	D_3	D_4	D_5
~~*Levenshtein*~~		1	1		
JaroWinkler ⇓			5		
MongeElkan ⇓	1	4		1	
$FN(\Sigma)$	1	5	6	1	

JaroWinkler de apenas 0,61. Como esta função foi a única utilizada na geração da árvore de decisão e o limiar de classificação era de 0,84 (vide Figura 8), este par foi incorretamente classificado como contatos distintos.

Analogamente, a Tabela 5 apresenta os casos de falha dos contatos distintos, ou seja, os falsos positivos. São apresentados os mesmos campos da Tabela 4.

Tabela 5: Casos de falha na identificação de contatos duplicados (falsos positivos).

Falha	D_1	D_2	D_3	D_4	D_5
~~*Levenshtein*~~	286	238	442	838	1354
~~*Jaccard*~~			637		
~~*JaroWinkler*~~		134			
~~*MongeElkan*~~			244		36
Jaccard ⇑	73	8		204	1375
JaroWinkler ⇑	332		236	1116	161
MongeElkan ⇑	11	323		18	
$FP(\Sigma)$	702	703	1559	2176	2926

Em D_1, a maioria das falhas são casos em que os escores retornados pela função *JaroWinkler* são maiores do que os esperados pelo modelo de classificação, totalizando 332 dos 702 casos (47,3%). Por exemplo, *JaroWinkler* (Adriana Gouveia, Adriana Jouris) = 0,93. Este e muitos outros pares de contatos distintos compostos por apenas dois nomes sendo que o primeiro nome é exatamente o mesmo retornam escores muito altos para esta função, justamente porque ela considera o tamanho do prefixo em comum.

Em D_5, os contatos *Fernando Luis Martins* e *Luis Fernando Tusnski* são detectados incorretamente como pares duplicados porque o escore retornado pela função de similaridade *Jaccard* é 0,5. Este valor é maior que o escore máximo de 0,33 esperado pelo modelo de classificação. Falhas deste tipo são as mais frequentes para este conjunto de dados (1375/2926 = 47%), junto dos casos em que não foi utilizada a similaridade entre os *e-mails* calculada pela função *Levenshtein* (1354/2926 = 46,3%).

6. CONCLUSÃO

Este trabalho apresentou um método para deduplicação de contatos que facilita o processo de integração e reduz consideravelmente o tempo em que um usuário levaria para associar manualmente contatos de diversas contas. Os experimentos realizados mostram que, utilizando funções de similaridade textual e aprendizagem de máquina, foi possível identificar corretamente até 92,1% dos pares de contatos duplicados que não compartilham números de telefones ou endereços de *e-mail*. Como estes pares não podem ser detectados por nenhuma das ferramentas apresentadas na Seção 2, fica evidente a contribuição do trabalho proposto quando comparado a estas ferramentas.

Entretanto, ainda podem ocorrer outros erros de identificação. Por exemplo, o contato com nome = "*Mãe*" armazenado no cartão SIM com o telefone residencial não seria detectado como duplicata do registro contendo o respectivo nome próprio e o número do celular. Ainda podem existir homônimos que não representam a mesma pessoa, como o caso de Orlando Marasciulo (registros 6, 7 e 8 da Figura 1).

Além de avaliar a qualidade dos algoritmos de agrupamento *Single Link* e *Click*, destaca-se como trabalho futuro a implementação de uma arquitetura em nuvem que armazene os modelos de aprendizagem locais e os integre gerando um modelo global. Os erros e acertos dos processos de deduplicação de cada usuário serão combinados de forma a aperfeiçoar o processo de deduplicação para todos.

O protótipo ainda será reimplementado como um serviço para que a cada inserção ou exclusão de um contato, a deduplicação seja feita de forma incremental e bastante eficiente. Além disso, será considerado o nono dígito nos números de telefone. Por fim, a interface gráfica servirá apenas para configuração de parâmetros e interação com o algoritmo de integração, onde o usuário poderá escolher entre duas ou mais representações do nome de um contato duplicado.

7. AGRADECIMENTOS

Parcialmente financiado pelos Programas Institucionais de Iniciação Científica, Tecnológica e de Inovação PROBIC/ FAPERGS, PIBIC-PIBITI/CNPq e PDE/FURG.

8. REFERÊNCIAS

[1] W. F. Ableson, R. Sen, C. King, and C. Ortiz. *Android em ação*. Campus, Rio de janeiro, 2012.

[2] A. Accaci. Duplicate contacts. `http://play.google.com/store/apps/details?id=com.accaci`, 2015. Acesso: julho de 2015.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48, 2003.

[4] E. N. Borges, K. Becker, C. A. Heuser, and R. Galante. A classification-based approach for bibliographic metadata deduplication. In *Proceedings of the IADIS Int. Conference WWW/Internet*, pages 221–228, 2011.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 313–324, 2003.

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

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI Workshop on Information Integration*, pages 73–78, 2003.

[8] P. Dabhi. Duplicate contacts delete. `http://play.google.com/store/apps/details?id=com.don.contactdelete`, 2015. Acesso: julho de 2015.

[9] G. Dal Bianco, R. Galante, M. A. Gonçalves, S. Canuto, and C. A. Heuser. A practical and effective sampling selection strategy for large scale deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2305–1319, 2015.

[10] M. G. de Carvalho, A. H. F. Laender, M. A. GonÇalves, and A. S. da Silva. Replica identification using genetic programming. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1801–1806, 2008.

[11] C. F. Dorneles, M. F. Nunes, C. A. Heuser, V. P. Moreira, A. S. da Silva, and E. S. de Moura. A strategy for allowing meaningful and comparable scores in approximate matching. *Information Systems*, 34(8):673–689, 2009.

[12] G. J. Kowalski and M. T. Maybury. *Information Storage and Retrieval Systems : Theory and Implementation*. Springer, Boston, MA, USA, 2002.

[13] R. Lecheta. *Desenvolvendo para iPhone e iPad*. Novatec, São Paulo, 2014.

[14] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 233–246, 2002.

[15] R. F. Machado, R. F. Pinheiro, E. A. Nunes, and E. N. Borges. Identificação de contatos duplicados em dispositivos móveis utilizando similaridade textual. In *Anais da Escola Regional de Banco de Dados*, p. 1–8, 2016.

[16] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[17] ORGwareTech. Contact merger. `http://play.google.com/store/apps/details?id=com.orgware.contactsmerge`, 2015. Acesso: julho de 2015.

[18] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, USA, 1993.

[19] A. M. Silva. Limpador de contatos. `http://play.google.com/store/apps/details?id=br.com.contacts.cleaner.by.alan`, 2012. Acesso: julho de 2015.

[20] D. M. Sunil. Duplicate contacts manager. `http://play.google.com/store/apps/details?id=com.makelifesimple.duplicatedetector`, 2014. Acesso: julho de 2015.

[21] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.