Contact Deduplication in Mobile Devices using Textual Similarity and Machine Learning

Eduardo N. Borges, Rafael F. Pinheiro and Graçaliz P. Dimuro
Centro de Ciências Computacionais, Universidade Federal do Rio Grande – FURG,
Av. Itália, km 8, 96203-900, Rio Grande, RS, Brazil

Keywords: Deduplication, Similarity, Mobile, Classification.

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 similarity functions, which scores are combined by a classification model based on decision trees, avoiding the need for an expert to manually configure similarity thresholds. The experiments show that the proposed method identified correctly up to 92% of duplicate contacts.

1 INTRODUCTION

Over the past few years, Internet and Web have changed the way people communicate. With the increase of the number of Web applications, users tend to accumulate many accounts in different services like email, social network, music and video streams and virtual stores. The progress of technology has provided access from mobile devices such as smartphones and tablets, to all services mentioned above. Managing information from many services or applications is a complex task for users. Some basic services of mobile devices can be affected by the redundancy of information automatically collected from different applications. For example, to find a contact on the contacts list with duplicates and incomplete information considerably reduces the productivity that a mobile device can offer.

Figure 1 shows a piece of a real contact list, including ten records, collected from different data sources represented by the icon on right side. Some information are already combined from some sources as shown in the record 3 (Google, Whatsapp and Waze). However, the records 4, 5 (Skype), 6 (Google+) and 8 (LinkedIn) represent the same person and could be integrated with the record 3 making the cluster D. This also happens in the cluster $A = \{1,2\}$ and $B = \{7,9\}$. The record 10 does not have a similar contact, so it remains alone.

Popular operating systems for mobile devices, like iOS and Android (Goadrich and Rogers, 2011), provide a contact association feature, but the user needs

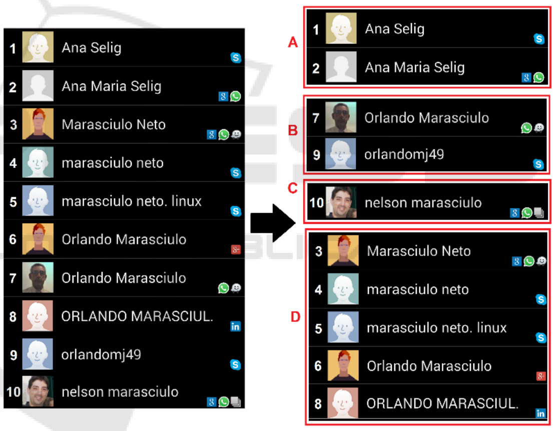


Figure 1: A list of contacts (left) and the expected result of deduplication (right).

to select the records he/she wants to combine. This association is stored in the device. If the user loses the device or reinstall the system, their contacts restored from an on-line account will not be associated.

This paper presents a deduplication method that uses some similarity functions to compare the fields that describe the contacts collected from multiple sources. The scores returned by these functions are used to train a classification model, based on decision trees, that may be used as part of an automatic contact association strategy (integration). The new method is compared to the strategies implemented by a set of applications for contact management available for free. The quality of the method is also experimentally evaluated on a real database with about 2000 records.

Borges, E., Pinheiro, R. and Dimuro, G.
Contact Deduplication in Mobile Devices using Textual Similarity and Machine Learning.
DOI: 10.5220/0006275100640072
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 64-72
ISBN: 978-989-758-247-9

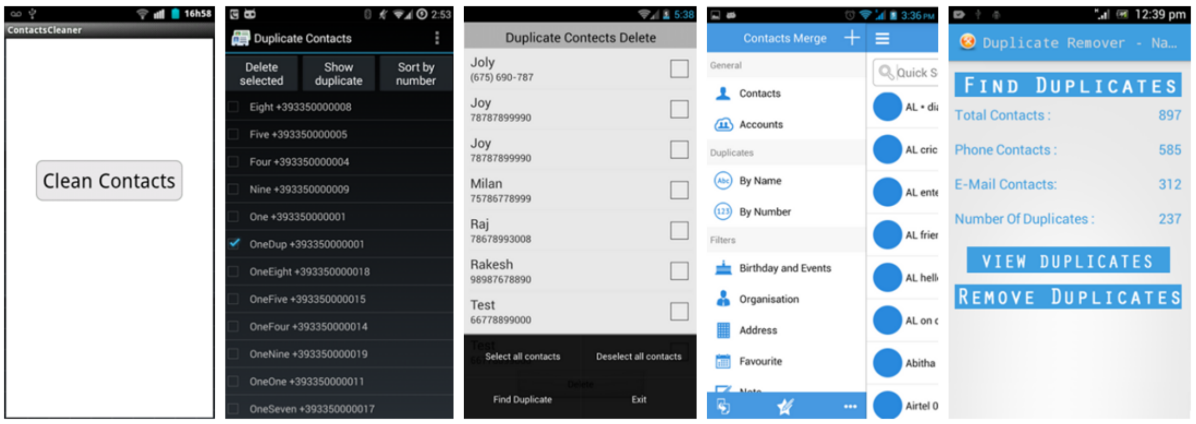


Figure 2: Graphical user interfaces of the apps (left to right): Contacts Cleaner, Duplicate Contacts, Duplicate Contacts Delete, Contact Merger and Duplicate Contacts Manager.

The rest of this paper is structured as follows. Section 2 presents a study on a set of five apps for contact management. Section 3 reviews studies of the scientific literature about fundamental concepts for the understanding of the proposed work. Section 4 specifies the proposed method to deduplicate contacts. The developed prototype and experimental evaluation results are discussed in Section 5. At the end, in section 6, the conclusions are presented and pointed out some future work.

2 CONTACT MANAGEMENT APPLICATIONS

On-line stores, like Google Play[1], Apple App[2], and Windows Phone Store[3], offer many applications for contacts management, however the majority of applications aims to make easier the insertion, edition, organization and sharing information about contacts. The user prefers these applications than the ones installed by default on the operating systems. There are few applications to identify and eliminate duplicate contacts.

A complete solution of data integration should indicate specific methods to perform the following tasks: extract data records from different and heterogeneous sources; transform the data to obtain a common representation, i.e. a compatible schema; identify semantically equivalent records which represent the same object; merge information from multiple sources; display to the user a set of records without duplicate information (Lenzerini, 2002). The studied applications perform only the last three tasks because they use methods available in the operating system to read the records in the same schema.

The Contact Cleaner app (Silva, 2012) removes duplicate records of the list of contacts comparing only the phone numbers. The graphical user interface shown in Figure 2 has a single button that when pressed removes duplicate contacts without any user interaction. After the duplicate contacts were removed, the app shows a notification with the number of excluded contacts. It is not possible to view the contacts detected as replicas and neither restore the original list of contacts.

Duplicate Contacts (Accaci, 2016) app allows to view duplicate contacts and to select the records that will be delete. A preliminary selection is automatically presented to the user using the equality of phone numbers, as shown in Figure 2. It is possible to configure a backup file with the last state of the contacts list before changes.

The third app studied is Duplicate Contact Delete (Dabhi, 2015). It has the same functions mentioned above, but also uses the contact's name to identify duplicates. The graphical interface is displayed in Figure 2. The three apps mentioned above are implemented exclusively for Android and they allow only eliminate duplicate contacts. Two apps that provide redundant information integration and association of contacts were also analyzed.

Contact Merge (ORGwareTech., 2015) is available for Android and Windows Phone. It combines all contacts' phone numbers with the same name, but keeps only one of the names for contacts with the same phone. This second strategy is dangerous because relevant information may be lost in the integration process. For example, the name of a contact can be replaced by a nickname, as in the case of in-

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

tegrating “Ben” and “Benjamin Baker”. The integration process can also remove an important workplace identifier in the contact description like the association between “Beatrice IBM” and “Beatrice”. The interface of this application is very attractive (see Figure 2) and allows to access other features in management of contacts, such as birthdays, addresses and favorite contacts.

Finally, Duplicate Contact Manager (Sunil, 2016) stands out because it also uses the email on deduplication. After detecting duplicate records, statistics are displayed on each type of contact and the number of duplicates found, which can be viewed or removed directly. Figure 2 (right) shows the graphical user interface displaying the feature mentioned above. Unfortunately, the integration feature is available only in the paid version and can not be tested. This application is only available for Android.

Table 1 summarizes the features of the studied apps and compares them to the method proposed in this paper. For each application, we present the fields used in deduplication, the comparison function of the fields, and the use of a machine learning strategy in the deduplication process.

The advantages and the main contributions of the this work are the following: (i) textual similarity functions, allowing to deal with many of the cases presented in the example in Figure 1 and (ii) a classification model which combines the scores returned by the similarity functions automatically, eliminating the need for an expert to set up similarity thresholds. The focus of this work is the method for identifying duplicate contact records. The integration and backup are not applicable. The Section 6 presents a full solution for merging contacts as a the future work.

3 DEDUPLICATION REVIEW

The task of identifying duplicate records that refer to the same real world entity is called deduplication (Borges et al., 2011). In recent years, different methods have been proposed for the deduplication of records, especially in the context of integration of relational data (Dal Bianco et al., 2015; Ogata and Komoda, 2013; Christen, 2012; Dorneles et al., 2009; de Carvalho et al., 2008; Chaudhuri et al., 2003; Bilenko and Mooney, 2003). Specific approaches for deduplicate contact records on mobile devices were not found in the literature.

Most of the proposed methods for identifying duplicate contacts uses the concept of measure of textual similarity, calculated by a similarity function or a distance function. The following subsections present the functions used in the proposed method (Peng and Mackay, 2014; Cohen et al., 2003). They were chosen because they cover different comparison strategies and are suitable to identify similarities in proper names. All functions return scores in the range $[0,1]$.

3.1 Jaccard

Let A and B be strings represented by sets of words. The function *Jaccard* calculates the similarity between A and B according to Equation 1. It returns the ratio between the number of words shared by strings and all the words that make up them, defined by:

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

3.2 Levenshtein

Let a and b be strings, The distance *Levenshtein* (*dist*) is defined as the minimum number of characters editions, as insertion, deletion, or replacement, necessary to transform a in b. The similarity is calculated as the complement of the normalized distance, defined by:

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

where $\max(a,b)$ is the length of the larger string.

3.3 JaroWinkler

Let m be the number of correlations between the characters and t the number of transpositions. The function *Jaro* (J), given by

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

calculates the similarity between a and b. *JaroWinkler* is a variation of *Jaro*, weighting the prefixes, with p size, of both strings, defined by:

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

3.4 MongeElkan

Let $A = \{a_1,...,a_K\}$ and $B = \{b_1,...,b_L\}$ be strings represented by sets of words K and L respectively. The function *MongeElkan* performs, for each pair of words, an auxiliary similarity function, in general *Levenshtein*, returning the average of the maximum similarities:

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

Table 1: Features of the related applications and the proposed work.

App	Fields	Comparison	Machine learning
Contacts Cleaner	phone number	equality	no
Duplicate Contacts	phone number	equality	no
Duplicate Contacts Delete	phone number, name	equality	no
Contact Merger	phone number, name	equality	no
Duplicate Contacts Manager	phone number, name, email	equality	no
proposed work	phone number, name, email	**similarity**	**yes**

4 CONTACT DEDUPLICATION

Deduplication can be a difficult task, mainly due to the following problems: use of acronyms, different formatting styles, distinct structure of metadata, variation in the representation of the content, omission of certain fields and omission of relevant content. In mobile devices it is not common to use acronyms to represent contacts' names and the data does not have a particular style. The schema of records is the same because the API of the operating systems can retrieve all the records in the same structure, even if it has been automatically collected from different social networks or other applications.

Therefore, the focus of contact duplication is to solve the problem of variation and omission of content, which is very frequent and more severe than in other contexts such as in digital libraries. While many duplicate contacts only share the first name, references have different representations of the authors (order of the names and abbreviations) and little change in the title of publications. They also have other relevant metadata, such as year, conference or journal. Since the vast majority of contacts only count on name and phone numbers, the deduplication method collects these two fields and generates an unique identifier for each record. The proposed method is divided into four main phases presented in Figure 3: acquisition and preprocessing, computation of similarities, classification of duplicate pairs and clustering equivalent contacts.

4.1 Collecting and Preprocessing

In the first phase the device contacts are collected from the internal memory, SIM cards and accounts linked to local or cloud applications, like messengers and social networks. For each imported contact, records with fields that represent name, phone numbers or emails are selected and stored.

The names are preprocessed by converting to lower case, removing accentuation and characters other than letters or numbers. The email login (without the domain) is stored in a new field.

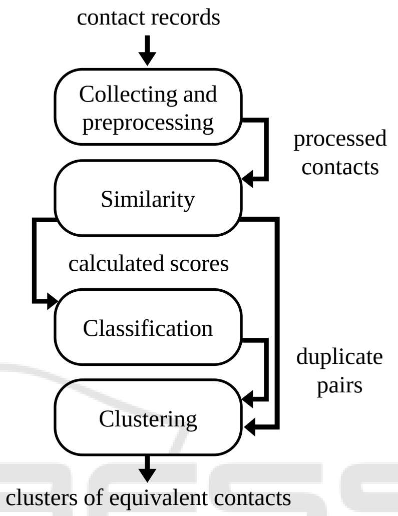


Figure 3: Phases of the proposed method for identifying duplicate contacts.

4.2 Similarity

In the second phase the records are combined in pairs. Those who share at least one phone number or e-mail (matching by equality) are directly identified as duplicate pairs and sent to the clustering phase.

The following similarity functions are applied on the other records and their fields:

- Levenshtein (email logins);
- Jaccard (names);
- JaroWinkler (names);
- MongeElkan (names).

Table 2 illustrates a pair of contacts and the scores returned by these similarity functions.

4.3 Classification

In the third phase, the scores returned by the similarity functions build new records, with an extra field that represents the pair's duplicity. These records are

Table 2: Pair of contacts and the scores returned by similarity functions.

Name	Email login	*Levenshtein*	*Jaccard*	*JaroWinkler*	*MongeElkan*
Mateus Gabriel Muller Mateus Muller	mateusmuller mateusmuller2	0,92	0,66	0,6	0,75

used as input to a classification model that labels each pair of contacts as *duplicate* or *distinct*. This model is a decision tree trained by C4.5 algorithm (Quinlan, 1993).

Each pair labelled as duplicate in this stage are added in the list of pairs previously identified with equal phone numbers or emails and they are sent to the clustering phase.

4.4 Clustering

Finally, in the fourth and last phase, the equivalent contacts can be grouped using two different strategies: (i) each record is similar to at least one record in the same cluster and (ii) all records of a cluster are similar to each other.

For implementing these strategies, we have defined a graph which each vertex represent a contact and the edges mean the duplicity. Two algorithms are performed over this graph (Kowalski and Maybury, 2002):

- Single Link - It returns a cluster for each connected component of the graph, implementing the first strategy;
- Click - It returns clusters representing complete subgraphs, implementing the second strategy.

Figure 4 shows the result of clustering algorithms considering the duplicate graph (left) as input.

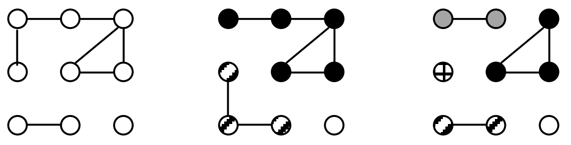
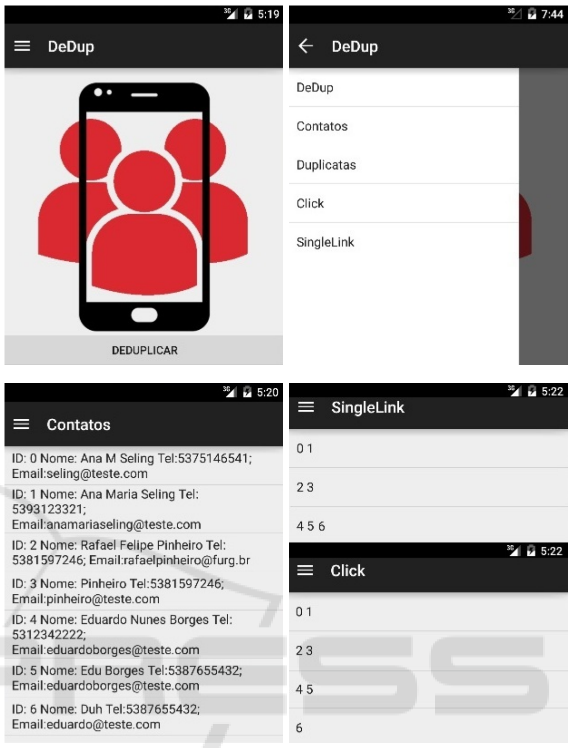


Figure 4: An example of *Single Link* (center) and *Click* (right) clustering strategies.

5 EXPERIMENTAL EVALUATION

This section describes the experiments conducted in order to validate the contacts deduplication method proposed in this paper. Figure 5 shows the interface of the prototype highlighting the home screen, menu features, a contacts list and clustering results grouping duplicate contacts using Single Link and Click algorithms. It was programmed in the Java language using the Android SDK.



5:20
Contatos

ID: 0 Nome: Ana M Seling Tel:5375146541; Email:seling@teste.com

ID: 1 Nome: Ana Maria Seling Tel: 5393123321; Email:anamariaseling@teste.com

ID: 2 Nome: Rafael Felipe Pinheiro Tel: 5381597246; Email:rafaelpinheiro@furg.br

ID: 3 Nome: Pinheiro Tel:5381597246; Email:pinheiro@teste.com

ID: 4 Nome: Eduardo Nunes Borges Tel: 5312342222; Email:eduardoborges@teste.com

ID: 5 Nome: Edu Borges Tel:5387655432; Email:eduardoborges@teste.com

ID: 6 Nome: Duh Tel:5387655432; Email:eduardo@teste.com

5:22
SingleLink

0 1

2 3

4 5 6

5:22
Click

0 1

2 3

4 5

6

Figure 5: Prototype implemented for Android, highlighting the results of Deduplication.

We used a real database, provided by a volunteer, with exact 1962 contacts imported from multiple data sources : internal memory, SIM card, Skype, Facebook, LinkedIn, Gmail and Google+. The quality of the method was evaluated using the percentage of correctly classified instances (accuracy) and the recall measure (Manning et al., 2008) to the class duplicate contacts. It was used the Weka[4] (Hall et al., 2009) tool for training and evaluation of classification models.

All records containing at least one name and a phone number or e-mail was selected. After the preprocessing, 1072 contacts remain valid. These contacts were combined two by two. Then 12 pairs with the same phone numbers or email were deleted (duplicate contacts detected by equality), totaling 574,044 pairs, among which only 66 represent duplicate contacts. For each pair, we executed the similarity functions presented above. The attributes relating to duplicity were added to the returned similarity scores, building the new records. We notice that these 12

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

easy-duplicates can be also detected by the majority of tools presented in Section 2. We focus in the remaining 66 duplicates which are hard to identify.

It was not possible to adopt an usual model evaluation strategy such as cross-validation because the classes are very unbalanced (66 duplicate pairs and 573,978 distinct). We have created five training sets with 1000 instances each. These instances were selected randomly, distributed as follows: 33 representing duplicate contacts (half of the 66 available) and 967 distinct. For each training set we have created a corresponding test set with the addition of instances available, i.e. containing the 573,044 remaining records, among which are present the other 33 pairs of duplicate contacts.

Figure 6 shows the distribution of values of each field (similarity scores) according to the class, considering one of the training sets. None of the similarity functions alone is able to properly separate the majority of duplicate contacts. In addition, an expert user would have great difficulty in assigning a threshold for each function and/or set a way to combine the scores appropriately. The results presented below show the contribution of using machine learning in the process of contact duplication.

5.1 Results

Table 3 summarizes the results of the trained models, i.e. the number of pairs of contact duplicate or distinct correct and incorrectly classified. For each data set (D) is presented: the frequency of true positive (TP), the frequency of false negatives (FN), the frequency of true negatives (TN), the frequency of false positives (FP), the percentage of total instances correctly classified (accuracy - Acc) and the recall of the class corresponding to duplicate contacts (R_{dup}), i.e. the percentage of duplicate pairs correctly identified. The last lines show the average and standard deviation.

The overall quality of the models can be seen by the high rate of test instances classified correctly. On average, 97% of the pairs of contacts have been identified in the correct class. When observed only duplicate classes were identified 27 to 33 pairs, resulting in an average of 30.4 (92.1%) duplicate pairs of contacts. The number of false negatives was significant only for datasets 2 and 3, which made the recall fall to 84.8 and 81.8% respectively.

Table 3: Deduplication results using the classifier C4.5.

D	TP	FN	TN	FP	Acc	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

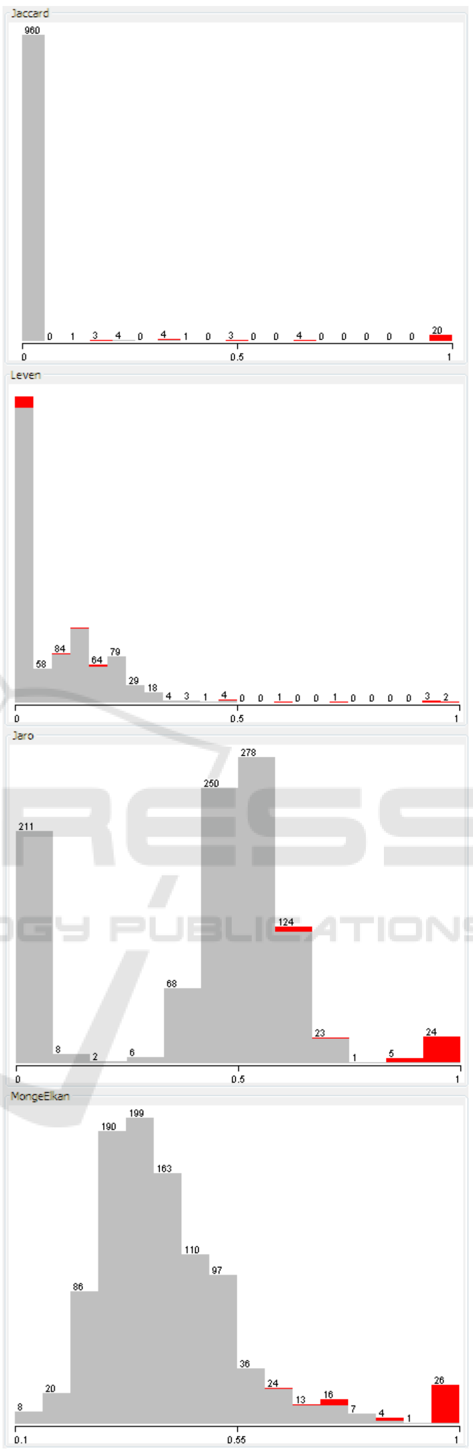


Figure 6: Distribution of similarity scores according to the class for one of the training sets.

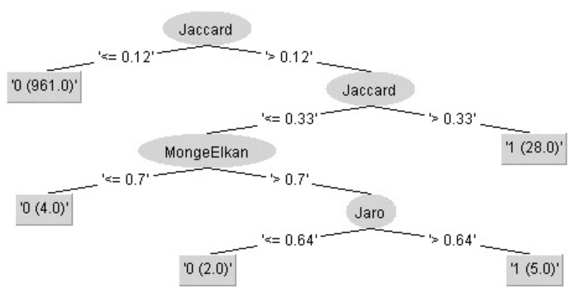


Figure 7: Model generated from the training set 1.

Despite the good results many different pairs were incorrectly classified as duplicates (1613 on average). The high standard deviation occurred because the 967 records randomly selected to compose the training sets are not sufficiently representative. However, the main goal of this work is to correctly classify cases where the contacts are duplicate. About false negatives, on average, only 2.6 of the 33 (7.9%) of duplicate contacts were classified incorrectly.

Figure 7 shows the model learned by C4.5 algorithm using training set 1. Class 1 refers to *duplicate contacts* and class 0 to *different contacts*. The root of the decision tree presents the most discriminatory attribute, indicating the importance of similarity between the contacts' names according to the *Jaccard* function. For scores less than or equal to 0.12, the model was able to classify 961 of the 967 (99.4%) distinct pairs available in the training set. For scores greater than 0.33, there were classified 28 of 33 (84.8%) duplicate contacts. The cases that were not identified by the first two nodes of the tree were classified by the leaf nodes that correspond to the scores returned by the similarity functions *MongeElkan* and *JaroWinkler*.

The lowest revocation R_{dup} = 81,8% presented in test set 3 can be explained by the simplicity of the model learned using the corresponding training set (Figure 8). The tree contains only one decision node that refers to the scores returned by the *JaroWinkler* function. For scores less than or equal to 0.84, all 967 different pairs available in the training set was correctly identified, but there was two classification errors of duplicate pairs. Values larger than 0.84 correctly identified 31 of 33 (93.9%) duplicate pairs.

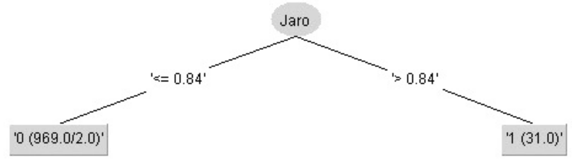


Figure 8: Model generated from the training set 3.

5.2 Analysis of the Failure Cases

Table 4 presents the cases in which the proposed method failed to identify the duplicate contact pairs, i.e. the false negatives. For each type of failure and dataset ($D_i | 1 \leq i \leq 5$) the number of cases is displayed. Failures are classification errors caused by: a similarity function not used in the model, represented by strikethrough, or a score returned by a given similarity function less ($\Downarrow$) or greater ($\Uparrow$) than the rule threshold that classify the evaluated instance. Some examples of records and the reasons why they fit into a particular type of failure are presented bellow.

In D_1, the pair of contacts whose names are "*Leonardo C3*" and "*Leonardo Emmendorfer*" is not identified because *MongeElkan* returns 0.5. This score is less than 0.7, so it was classified by a leaf node as distinct contacts (see Figure 7). This case of failure is very difficult to solve because *C3* is a qualifier that refers to the workplace of the contact, while *Emmendorfer* is the surname.

In D_3, although *MongeElkan* returned the maximum score for contacts "*Nyland*" and "*Nathan Nyland*", *JaroWinkler* returns only 0.61. Due to this function was the only one used in the decision tree and the classification threshold was 0.84 (see Figure 8), this pair was incorrectly classified as distinct contacts.

In the same way, Table 5 shows the cases of failure to identify distinct contacts, i.e. false positives. It presents the same fields as in Table 4.

Table 4: Cases of failure to identify duplicate contacts (false negatives).

Failure	D_1	D_2	D_3	D_4	D_5
~~Levenshtein~~		1	1		
JaroWinkler $\Downarrow$			5		
MongeElkan $\Downarrow$	1	4		1	
$FN(\Sigma)$	1	5	6	1	

Table 5: Cases of failure to identify distinct contacts (false positives).

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

In D_1, most failures are cases where the scores returned by the *JaroWinkler* function are higher than expected by the classification model, totaling 332 of the 702 cases (47.3%). For example, *JaroWinkler* (Adriana Gouveia, Adriana Jouris) = 0.93. This and many other distinct pairs of contacts consisting of only two names and where the first name is exactly the same return very high scores for this function. This behavior was because *JaroWinkler* considers the size of the prefix in common to the strings.

In D_5, the contacts *Fernando Luis Martins* and *Luiz Fernando Tusnski* are incorrectly detected as distinct pairs because the score returned by the *Jaccard* similarity function is 0.5. This value is higher than the maximum score of 0.33 expected by the classification model. This type of error was most frequent for this dataset (1375/2926 = 47%), in addition to the cases where the similarity between emails was not used by the *Levenshtein* function (1354/2926 = 46.3%).

6 CONCLUSION

This paper presented a method for deduplication of contacts that facilitates the integration process and considerably reduces the time a user would take to manually associate contacts from different accounts. The experiments show that, using textual similarity functions and machine learning algorithms, it was possible to correctly identify up to 92.1% of duplicate contacts pairs that do not share telephone numbers or e-mail addresses. The contribution of the proposed work when compared to the tools presented in Section 2 becomes evident because these pairs of contacts can not be detected by any of them.

However, other identification errors can still appear. For example, the contact with name = "*Mom*" stored on the SIM card with the home phone number would not be detected as a duplicate of the record containing its proper name and cell number. There may still be homonyms that do not represent the same person, as in the case of "*Orlando Marasciulo*" (records 6, 7 and 8 of Figure 1).

As future work we intend to adopt a cloud architecture that stores the users local learning models and integrates them generating a global model. The hits and errors of the deduplication processes will be combined in order to improve the deduplication process for all users. Also, the prototype will be reimplemented as a service allowing an efficient incremental deduplication for each insertion or deletion of a contact.

Finally, the graphical interface will only allow the user set up parameters and interact with the integration algorithm, choosing between two or more representations of a duplicate contact name.

REFERENCES

Accaci, A. (2016). Duplicate contacts, v. 3.23. http://play.google.com/store/apps/details?id=com.accaci. Available: November, 2016.

Bilenko, M. and Mooney, R. J. (2003). 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.

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

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

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

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

Dabhi, P. (2015). Duplicate contacts delete, v. 1.1. http://play.google.com/store/apps/details?id=com.don.contactdelete. Available: November, 2016.

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

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

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

Goadrich, M. H. and Rogers, M. P. (2011). Smart smartphone development: Ios versus android. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 607–612, New York. ACM.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

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

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *Proceedings of the ACM*

SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pages 233–246.

Manning, C. D., Raghavan, P., and Schutze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Ogata, M. and Komoda, N. (2013). The parameter optimization in multiple layered deduplication system. In *Proceedings of the International Conference on Enterprise Information Systems*, volume 2, pages 143–150.

ORGwareTech. (2015). Contact merger, v. 3.8. http://play.google.com/store/apps/details?id=com.orgware.contactsmerge. Available: November, 2016.

Peng, T. and Mackay, C. (2014). Approximate string matching techniques. In *Proceedings of the International Conference on Enterprise Information Systems*, volume 1, pages 217–224.

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

Silva, A. M. (2012). Contacts cleaner, v. 1.6. http:// play.google.com/store/apps/details?id=br.com.contacts.cleaner.by.alan. Available: November, 2016.

Sunil, D. M. (2016). Duplicate contacts manager, v. 2.8. http://play.google.com/store/apps/details?id=com.makelifesimple.duplicatedetector. Available: November, 2016.