Using Trainable Duplicate Detection for Automated Public Data Refining

Martin Lipták*
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
mliptak@gmail.com

Abstract

Public institutions share important data on the Web. These data are essential for public investigation and thus increase transparency. However, it is difficult to process them, since there are numerous mistypings, disambiguities and duplicates. In this paper we propose an automated approach for cleaning of these data, so that further querying result is reliable. We develop a duplicate detection method that can train weights of features on small amount of training samples and then predict duplicates on the rest of data. We evaluate our method on two real-world data sets.

Categories and Subject Descriptors

H.2.5 [**Information Systems**]: Database Management—*Heterogeneous Databases*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

Keywords

data integration, duplicate detection, machine learning

1. Introduction

Public institutions have legal obligations to share certain data on the Web. While public registers (e.g. businesses, organizations) and bulletins (public procurements) are essential for business communication, other data increase transparency of public institutions and enable public investigation (public contracts). Despite the fact that these data are becoming publicly available on the Web, there are two problems.

The first problem is format and structure that might not be suitable for machine processing. For example some documents are published as scanned images with censored names and prices. This makes such documents difficult to investigate by a human expert and almost impossible to process with computer. For example company liquidations in Slovakia are published in periodic PDF bulletins as unstructured text content and it is difficult to reliably find out if a company is being liquidated or the liquidation is being cancelled. Fortunately the most common format is HTML, which is easy to parse and in most cases provides structure. Moreover, the state of public data is nowadays slowly being improved.

The second problem are various mistypings, disambiguities and duplicates. They are common even in correctly parsed and structured data. A few examples:

- Name of a person might be typed correctly in one business register extract and with a mistake in another one. We can use different fields in these entities like address and find out whether they refer to the same person.
- Two entities have equal names, but different addresses. How do we know if the person has moved or these two entities refer to different people? We can use a heuristics that when two entities with equal names and different addresses occur in the same company, these entities are duplicates.
- Two entities have both names and addresses equal. We need to find out if they are really the same person or father and son living at the same address.
- Two names might differ only in academic degrees. For example one inludes Ing., the other one does not. When addresses are equal, these entities might refer to the same person.
- Two addresses are equal except that one address contains Bratislava - Dúbravka, the other one only Bratislava. If the names are equal, these entities should probably refer to the same person.

In this paper we address the second problem by developing trainable duplicate detection method. Our method

*Bachelor degree study programme in field Informatics. Supervisor: Ing. Ján Suchal, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, STU in Bratislava.
Work described in this paper was presented at the 8th Student Research Conference in Informatics and Information Technologies IIT.SRC 2012.

Lipták, M. Using Trainable Duplicate Detection for Automated Public Data Refining. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 4, No. 2 (2012) 51-55

cleans off (or refines) duplicates, mistypings and other disambiguities to make data querrying result consistent and reliable.

2. Trainable duplicate detection

We propose a duplicate detection approach based on supervised machine learning algorithms [11]. We use a logistic regression classifier [5] and Support Vector Machine (SVM) classifier [2] to predict whether samples are duplicates or not. The classifier trains weights of features, provided by user for particular database (like Levenshtein distance of compared fields or presence of particular combination of substrings in compared fields). The user also provides a labeled set of samples that is used to train the classifier. Trained classifier can detect duplicates by predicting using learned feature weights.

3. Experiment 1

We have evaluated our method on a database of people occurring in Business Register of the Slovak Republic provided by foaf.sk. There are many duplicates and it is difficult to determine, who exactly occurs in which company.

3.1 Data Set

We have chosen 4,298 out of 569,999 total people occurring in slovak business register in June 2011. We selected mostly duplicate records, though we have added many individual records as well. Our samples are all possible pairs of selected records (omiting the same pairs with reordered records and including pairs of two same records), therefore we have $4,298 * (4,298 + 1)/2 = 9,238,551$ samples. Each sample consists of two records and each record provides name and address attributes. We have extracted academic degrees out of names and compare them individually. 80% of samples have been used as a training set and 20% as a test set for final precision-recall evaluation.

3.2 Features

For training duplicate detection we used combinations of following features (feature sets):

- **Label.** A feature equal to sample's label; 1 for duplicates and 0 for non-duplicates.
- **Equal names.** This feature is 1, when compared names are equal, otherwise 0.
- **Equal addresses.** This feature is 1, when compared addresses are equal, otherwise 0.
- **Levenshtein distance of names.** This feature represents Levenshtein distance[1] of compared names.
- **Levenshtein distance of address.** This feature represents Levenshtein distance of compared addresses.
- **N-gram similarity of names.** This feature represents N-gram similarity[2] of compared names.
- **N-gram similarity of addresses.** This feature represents N-gram similarity of compared addresses.
- **Combination of academic degrees.** We have created $N * (N + 1)/2$ features for all pairs of N academic degrees occurring in sample names. For a given sample, degree combinations occurring in compared names are 1, all the others are 0. This is based on assumption that two people with the same name, however, first of them is Ing. and the second MUDr., are probably not duplicates.
- **Disjunction of academic degrees.** We have created N features for all academic degrees occurring in sample names. For a given sample, feature for particular degree is 1 when the degree occurs in exactly one of the compared names. If the degree occurs in none of them or in both, the feature is 0. This is based on assumption that presence of some degrees (like ml. or st., which mean junior and senior respectively when concerning father and son) in one record and absence in the other one indicates that they are not duplicates.

3.3 Methodology

There is a set of heuristics already detecting duplicates on foaf.sk. We have used their results for training and as a baseline for measuring precision, recall and F_1 score. Precision represents detected duplicates that are actual duplicates (is decreased by false positives) and recall represents actual duplicates that were detected (is decreased by false negatives). Both metrics should preferably be as high as possible and reasonably balanced. F_1 score combines both metrics (equations 1 - 3).

$$precision = \frac{true\,positives}{true\,positives + false\,positives} \quad (1)$$

$$recall = \frac{true\,positives}{true\,positives + false\,negatives} \quad (2)$$

$$F_1 = 2 * \frac{prescision * recall}{precision + recall} \quad (3)$$

3.4 Results

Table 1 shows performance of various feature sets on test set. Overall high F_1 score is caused by the nature of our samples, as they are a cartesian product of individual records and the vast majority of them are not duplicates (for example John Doe record is compared to John Doe, John Doa and Ing. John Doe and 3 duplicates are found, but the other 2 milion comparisons are not duplicates).

Simple name and address equality comparison yields result of 142 false positives and 13 false negatives. False positives are caused by records with equal both names and addresses that were not labeled as duplicates by baseline duplicate detection. This can be explained by either lack of other features besides names and addresses, like company occurrences, or more probably by errors in baseline duplicate detection. False negatives are mostly typos that were not detected.

Levenshtein distance decreases number of false negatives, because it accepts misspelled strings. However, it dramatically increases number of false positives, as simple edit

[1] Levenshtein distance (or edit distance) of two strings computes number of edit operations (insertion, substitution, deletion) required to change the first string into the second one.

[2] N-gram similarity is a string similarity metrics as described in [10]. Sets of N-grams (that is tokens of size N characters) are created from compared strings. N-gram similarity is a Jaccard similarity of these sets.

operations count might not do a good job in distinguishing between a typo and similar spelling (Martin Lipták vs. Martin Lupták) and it does not take degrees into account.

N-grams of names without degrees and addresses yield same results as equality comparison regardless of N. In contrast, N-grams of names with inclusion of degrees decrease false positives, as they consider names with different degrees non-duplicates, but increase false negatives, because many names differing only in degrees are actual duplicates.

Degree combinations and disjunctions decrease number of false positives of individual Levenshtein distance, because they include degrees in comparisons. Inspection of learned weights of degree features revealed that disjunction of bc. (0.51), csc. (2.89), ing. (4.09), judr. (1.32) and mgr. (1.75) and a combination if judr. - mgr. (1.57) rise probability of records being duplicates

Figure 3.4 shows a learning curve of degree combinations. Curves of other feature sets look similar. Training set size is represented by x-axis (we have tried values 5%, 10%, 20%, 40%, 60% and 90%), misclassification error by y-axis. Cross-validation set size is constantly 100%. With growing training set size, cross-validation error slightly decreases, but training set error takes off. Both curves will eventually converge, but error tends to remain stable. This shows a problem of high bias. Further work needs to be done not in the amount of new samples, but in better features.

4. Experiment 2

We have used data set parsed from Registers and Evidences of Ministry of Interior of the Slovak Republic. It contains organizations and occurrences of people in them. It is difficult to determine who occurs in which organization, as there are many duplicates. Methodology and many features are same as those used in the first experiment.

4.1 Data Set

In contrast to the first experiment, our data set has been manually labeled. We aimed for a balanced (a few examples of many situations that can occur) and consistently-labeled sample selection. The size of data set is 250 people (or records). Our samples are again all possible pairs of selected records (omiting the same pairs with reordered records and including pairs of two same records), therefore we have $250 * (250 + 1)/2 = 62750$ samples. Each sample consists of two records and each record provides name and address attributes. In addition to extraction of academic degrees out of names, we have split addresses into parts and added a entity-relation heuristics to each sample (like existence of common organization for both compared people). 80% of samples have been used as a training set and 20% as a test set for final precision-recall evaluation.

4.2 Features

We have added following new features (feature sets):

- **Common organization.** Heuristics is 1 when compared people occur in a common organization, 0 otherwise. We assume that if two people with similar names have completely different addresses and they occur in a common organization, they are probably duplicates.
- **Empty address.** Heuristics is 1 when one of campared addresses is missing, 0 otherwise. This is possible in Registers and Evidences, despite it was not the case of Business Register. We have used this feature to balance the case, when a missing address causes high Levenshtein distance or zero similarity of compared addresses.

4.3 Results

Tables 2 and 3 show performance of various feature sets on test set using a logistic regression and SVM classifier respectively.

First 3 logistic regression comparisons do not include academic degrees. Degree combinations and disjunctions were expected to deal with degrees instead of comparing them along with names. However, they did not decrease number of errors at all. The data set of 250 records could not provide enough samples for these features to train.

Logistic regression classifier could not adapt well for heuristics at first. Therefore we have tried a SVM classifier. SVM provided better results for heuristics. Further inspection of errorous examples has shown that heuristics have solved some of complicated examples, but they have also introduced new errors.

Splitting address into parts did not influence number of errors in a significant way. It even increased number of errors for some cases. Levenshtein distance and N-gram similarity can deal with full addresses themselves.

Despite adding new features for more complicated examples, learning curves of both logistic regression and SVM looked similar to the learning curve on figure 3.4. New features didn't help classifiers adapt to more complicated examples.

5. Related work

Misspellings, disambiguities and duplicates are a major problem not only in the public data domain. In fact every database possibly merged of multiple sources needs to be cleaned so that queries provide reliable results. This process is widely known as data integration, duplicate detection or record linkage. [6, 12, 7]

M. Bilenko and R. Mooney propose to employ learnable string distance functions for duplicate detection task. They have used Levenshtein distance with contiguous sequences of mismatched characters called affine gaps. Expectation maximization algorithm trains parameters for affine gap penalties for individual fields. Support vector machine classifier trains equality of compared records. This method is based on notion that different edit operations have varying significance in different domains. For example a digit substitution in a street address makes a major difference, because it effectively changes the house number, but a letter substitution is more likely caused by a typo or an abbreviation. [1]

W. Cohen and M. Richman have developed an adaptive and scalable method based on logistic regression. Weights are trained for particular domain from a labeled data set. Their method overcomes two other non-adaptive base-

Feature set	FP	FN	F_1 score
=(labels)	0	0	1
=(names), =(addresses)	142	13	0.9293
L(names), L(addresses)	326	3	0.9318
2G(names), 2G(addresses)	142	13	0.9293
3G(names), 3G(addresses)	142	13	0.9293
4G(names), 4G(addresses)	142	13	0.9293
5G(names), 5G(addresses)	142	13	0.9293
6G(names), 6G(addresses)	142	13	0.9293
2G(names with degrees), 2G(addresses)	138	40	0.9177
3G(names with degrees), 3G(addresses)	138	46	0.9147
4G(names with degrees), 4G(addresses)	136	50	0.9135
5G(names with degrees), 5G(addresses)	135	53	0.9124
6G(names with degrees), 6G(addresses)	135	54	0.9119
L(names), L(addresses), degree combinations	135	39	0.9194
L(names), L(addresses), degree disjunctions	135	23	0.9274

Table 1: Performance of various feature sets on test set. Abbreviations: equality (=), levenshtein distance (L), n-gram similarity (nG), false positives (FP), false negatives (FN).

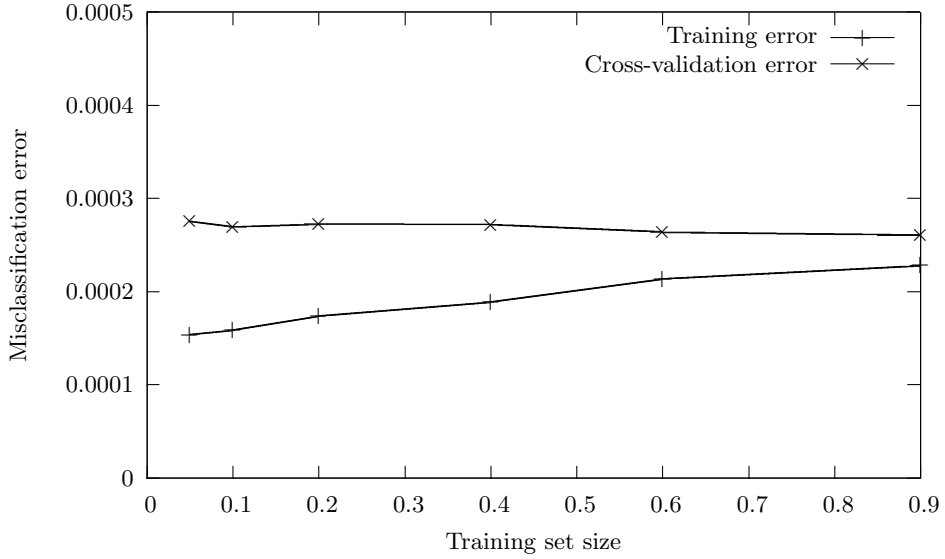

Figure 1: Learning curve of degree combinations

Feature set	FP	FN	F_1 score
L(names), L(addresses)	2	8	0.9728
L(names), L(addresses), degree combinations	2	8	0.9728
L(names), L(addresses), degree disjunctions	2	8	0.9728
=(names with degrees), =(addresses)	7	12	0.9485
L(names with degrees), L(addresses)	4	13	0.9534
2G(names with degrees), 2G(addresses)	3	9	0.9674
3G(names with degrees), 3G(addresses)	3	10	0.9646
=(names with degrees), =(address parts)	2	11	0.9644
L(names with degrees), L(address parts)	2	9	0.97
2G(names with degrees), 2G(address parts)	1	10	0.9697
3G(names with degrees), 3G(address parts)	1	10	0.9697
L(names with degrees), L(address parts), heuristics	3	8	0.97

Table 2: Performance of various feature sets on test set using a logistic regression classifier. Abbreviations: equality (=), levenshtein distance (L), n-gram similarity (nG), false positives (FP), false negatives (FN).

Feature set	FP	FN	F_1 score
=(names with degrees), =(addresses)	7	12	0.9485
L(names with degrees), L(addresses)	2	13	0.9587
2G(names with degrees), 2G(addresses)	2	10	0.9672
3G(names with degrees), 3G(addresses)	2	11	0.9644
L(names with degrees), L(addresses), heuristics	2	8	0.9728
=(names with degrees), =(address parts)	2	17	0.9470
L(names with degrees), L(address parts)	4	11	0.9591
2G(names with degrees), 2G(address parts)	6	8	0.9623
3G(names with degrees), 3G(address parts)	6	10	0.9568
L(names with degrees), L(address parts), heuristics	2	7	0.9756

Table 3: Performance of various feature sets on test set using a SVM classifier. Abbreviations: equality (=), levenshtein distance (L), n-gram similarity (nG), false positives (FP), false negatives (FN).

line methods and is almost as good as the third baseline method. [4]

D. Kalashnikov and S. Mehrotra in contrast to the traditional way of data cleaning, when only entity attributes are used to differentiate between entities (equality or similarity of selected attributes), propose inclusion of relations between compared entities. Firstly they use an attribute-based approach for quick candidate selection. Then graph-based techniques are used to discover relations that exist among these entities. They have shown a significant improvement on two real-world data sets. [9]

W. Cohen and other authors propose a name comparison toolkit that can help an expert with database integration. Toolkit offers many string comparison metrics like Levenshtein distance, TF-IDF, Jaro, Monge-Elkan or Jensen-Shannon. [3]

M. Hernández and S. Stolfo have developed a method for removing duplicates from databases of 100 milion to 1 bilion records in a matter of days. The first phase of quick sorting groups similar records together, more expensive rule-based comparisons in a sliding window across similar records are executed afterwards. [8]

6. Conclusion and future work

We have proposed a trainable duplicate detection method which uses user-provided labeled data set and feature sets. We have developed a prototype and evaluated it on two real-world data sets. Results of the first experiment have shown promissing results for simple duplicate detection examples. Further features added in the second experiment were supposed to improve results of our method for more complicated examples. Moreover, a manually labeled data set was used in the second experiment for more reliable evaluation. Splitting addresses into parts did not help significantly, as data set addresses were mostly already normalized and string metrics could deal with small differences themselves. Features for academic degrees could not adapt for small feature set used in the second experiment. Inclusion of entity relations improved results and has shown, that considering relations among entities besides their individual attributes is important for data cleaning.

We find machine learning-based approach for cleaning of Business Register and Registers and Evidences of Ministry of Interior unsuitable. We suppose that non-adaptive data normalization and pairing pipeline would clean complicated examples that our adaptive method could not and therefore would yield better results. The pipeline would leverage user-provided functions for particular database instead of labeled training samples. This approach should be evaluated in future efforts to clean Slovak public data.

Acknowledgements. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0971/11.

References

[1] M. Bilenko and R. Mooney. Employing trainable string similarity metrics for information integration. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 67–72, 2003.

[2] C.-c. Chang and C.-j. Lin. A Library for Support Vector Machines. *Library*, 2(3), 2011.

[3] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. *Learning*, 2003.

[4] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480, 2002.

[5] R.-e. Fan, K.-w. Chang, C.-j. Hsieh, X.-r. Wang, and C.-j. Lin. LIBLINEAR: A Library for Large Linear Classification. *Corpus*, 9(2008):1871–1874, 2012.

[6] I. P. Fellegi and A. B. Sunter. Record Linkage, 1969.

[7] L. Gu, R. Baxter, D. Vickers, and C. Rainsford. Record Linkage: Current Practice and Future Directions. *Science*.

[8] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 1998.

[9] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems*, 31(2):716–767, June 2006.

[10] G. Kondrak. N-gram similarity and distance. *String Processing and Information Retrieval*, pages 115–126, 2005.

[11] S. B. Kotsiantis. Supervised Machine Learning : A Review of Classification Techniques. *Informatica*, 31(3):249–268, 2007.

[12] W. Winkler. The state of record linkage and current research problems. *Statistical Research Division, US Census Bureau*, 1999.