A Distance Measure for Approximate Joins over Residential Addresses

Free University of Bolzano/Bozen
Faculty of Computer Science
Bachelor of Science in Applied Computer Science

Student: Davide Schuen

Supervisior: Johann Gamper
Tutor: Nikolaus Augsten

Academic Year 2003/2004
2nd Graduation Session - October 29, 2004

October 14, 2004

Abstract

The Municipality of Bolzano-Bozen has databases that store different information about the citizens. Data about a single object is distributed over several data sources. For some administrative tasks the data inside these heterogeneous and autonomous databases has to be joined. Applying standard joining techniques ends up with poor results, as there are no common keys. The data shows irregularities because of spelling mistakes, abbreviations and different naming conventions in the data fields. This work proposes a distance measure on common attributes of the databases, the addresses and the personal data. This measure is used to approximately join tuples coming from different databases.

In this thesis I analyze the accuracy of the matches by evaluating a number of experiments executed on the databases of the cadastre, registration office and the electric power company. A tool generates the experiments applying the distance measure on residential and personal data of two tuples using q-grams and edit distance algorithms. The data is cleaned to improve the quality and quantity of good approximate matches.

Contents

1 Introduction

This introduction section will provide the necessary background information on the topic and then state the actual definition of the project. At the end a brief outline of the contents will give insight into the structure of this thesis.

A typical situation found in a public administration environment is a number of autonomous databases which store information about the same real-world objects. The Municipality of Bolzano-Bozen has many autonomous databases, where the content of the databases is very similar, but not equal. For example, the registration office, the cadastre, and the electric power company all store data about citizens and apartments. Although they speak about the same objects, they are usually interested in different relations between them, e.g. the registration office stores who lives in an apartment, while the electric power company is interested in who pays the electricity bill for it. For several tasks, these databases have to be joined. Searching corresponding entries in different databases manually is tedious and often not feasible. Joining the addresses with standard join techniques (SQL-join) ends up with very poor results, as only exact matches are found. The problem hereby is, that some databases do not have a common key, that could be used as a join attribute, as the databases are referenced with a different granularity level. The most common attributes found in the tables of the databases are the residential addresses and the personal data. So, to get virtually one database, there has to be done a join on addresses and personal data [1].

The project not only means implementing a piece of software, but rather consists also of the whole analysis, planning and evaluation work that needs to be carried out. Therefore this thesis shall also describe the theoretical background of the project realization by providing insight in the whole development process, also revealing the difficulties faced and corresponding problem solutions proposed. Section two illustrates the problem description at the Municipality and lists the objectives in the project. Section three describes the string matching algorithms used to match approximately the data records of the different databases. Section four lists all steps necessary for joining the databases. It describes step by step the work that has to be done to get the correspondent records as well as details in the implementation. Section five gives an insight on the implementation of the two developed applications relying on the different acquired matching techniques. Section six contains the experiments of the calculated results as well as an accuracy analysis of the found matches. Section seven draws some conclusion and proposes further improvements.

2 Problem Description

We want to match correspondent records of different databases at the Municipality of Bolzano. The databases used for the matching are the databases of the cadastre, registration office and electric power company.

2.1 Data at the Municipality of Bolzano

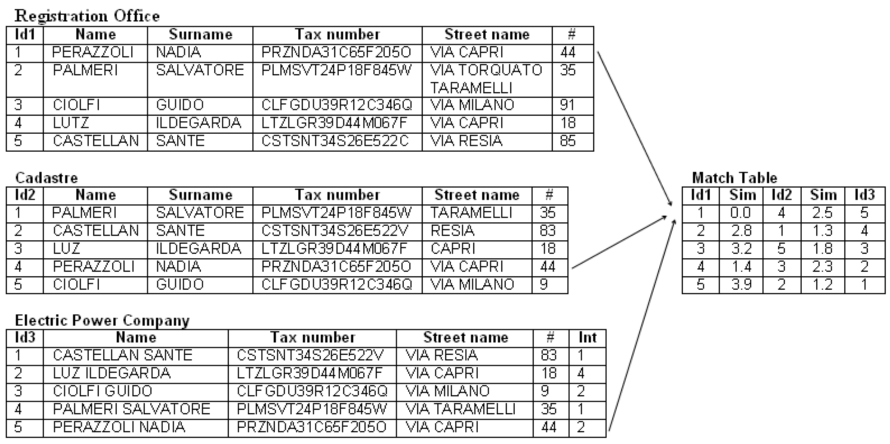
In our project we consider 3 databases used by the the Municipality of Bolzano-Bozen to provide their services to the citizens. This databases are the cadastre, the registration office and the electrical power company. The Municipality want to have all relevant data about each address. The goal is to have a map which acts as an interface to all addresses. With help of this interface one is able to extract relevant data from different databases for a particular building in the city. We have to build a matching table which matches correspondent records in different tables (figure 1 on page 3).

Example: *Mrs. Palazzetto Maria is registered at the registration office. She is born on the 25.05.1967 in Bolzano, lives in Viale Druso 23, apartment number 2, floor 3. We find the correspondent record in the cadastre. We get additional information on the building where she lives. The cadastre tells us that Mrs. Palazzetto Maria born on the 25.05.1967 in Bolzano lives in Druso 23, apartment number 2, parcel of land 1034 registered on the 19.04.1980. We match this data with the electric power company to gather information about the person which pays the electricity bill for this apartment. The correspondent match demonstrates that Palazzetto Carlo, born on the 12.02.1947 in Trento pays the electricity bill for the apartment in Viale Druso 23, apartment number 2, internal 1, floor 3.*

2.2 Objectives

The main objective of the thesis is to match records of three tables. The distance measure is computed using address and residential data as joining attributes. The matching is applied on the autonomous databases of the cadastre, registration office and the electric power company.

I have implemented two tools: the first program extracts the data out of the text files to make it suitable for a relational database. The second tool cleans and matches the data with approximate string matching algorithms. An analysis on the calculated results is done to estimate the reliability and exactness of the results.


Registration Office

Id1	Name	Surname	Tax number	Street name	#
1	PERAZZOLI	NADIA	PRZNDA31C65F205O	VIA CAPRI	44
2	PALMERI	SALVATORE	PLMSVT24P18F845W	VIA TORQUATO TARAMELLI	35
3	CIOLFI	GUIDO	CLFGDU39R12C346Q	VIA MILANO	91
4	LUTZ	ILDEGARDA	LTZLGR39D44M067F	VIA CAPRI	18
5	CASTELLAN	SANTE	CSTSNT34S26E522C	VIA RESIA	85

Cadastre

Id2	Name	Surname	Tax number	Street name	#
1	PALMERI	SALVATORE	PLMSVT24P18F845W	TARAMELLI	35
2	CASTELLAN	SANTE	CSTSNT34S26E522V	RESIA	83
3	LUZ	ILDEGARDA	LTZLGR39D44M067F	CAPRI	18
4	PERAZZOLI	NADIA	PRZNDA31C65F205O	VIA CAPRI	44
5	CIOLFI	GUIDO	CLFGDU39R12C346Q	VIA MILANO	9

Electric Power Company

Id3	Name	Tax number	Street name	#	Int
1	CASTELLAN SANTE	CSTSNT34S26E522V	VIA RESIA	83	1
2	LUZ ILDEGARDA	LTZLGR39D44M067F	VIA CAPRI	18	4
3	CIOLFI GUIDO	CLFGDU39R12C346Q	VIA MILANO	9	2
4	PALMERI SALVATORE	PLMSVT24P18F845W	VIA TARAMELLI	35	1
5	PERAZZOLI NADIA	PRZNDA31C65F205O	VIA CAPRI	44	2

Match Table

Id1	Sim	Id2	Sim	Id3
1	0.0	4	2.5	5
2	2.8	1	1.3	4
3	3.2	5	1.8	3
4	1.4	3	2.3	2
5	3.9	2	1.2	1

Figure 1: Match table and correspondent records in the tables

Exact joins on the address attributes give poor results. Only few correspondences can be found using standard joining techniques on residential data.

```
SELECT T1.*, T2.*
FROM T1, T2
WHERE T1.address = T2.address
```

By applying approximate join techniques on residential and personal data of two tables, we build a *matching table M* where we save the key values of the correspondent records. We can take advantage of the matching table to get the correspondent records in the tables. *id1 and id2* are the key value of the tables. *M(id1, id2)* is the match table containing correspondent records.

```
SELECT T1.*, T2.*
FROM T1, T2, M
WHERE T1.id1=M.id1 AND T2.id2=M.id2
```

3 Approximate String Matching Techniques

In this section I am going to describe, which approximate string matching algorithms were used and how they work. For this project I use 2 approximate string matching techniques, namely q-gram and weighted Levenshtein distance (shortly edit-distance).

Let Σ be a finite alphabet of size $|\Sigma|$. I use lowercase Greek symbols, such as α, possibly with subscripts, to denote strings in Σ^*. Let $\sigma \in \Sigma^*$ be a string of length n. I use $\sigma[i...j]$, $1 \leq i \leq j \leq n$, to denote a substring of σ of length j -i+1 starting at position i. λ denotes the *empty string.*

3.1 Weighted Levenshtein Distance

The *Levenshtein distance* - also known as *edit distance* between two strings is the minimum number of edit operations (i. e., *insertions, deletions,* and *substitutions*) of single characters needed to transform the first string into the second [2].

An edit operation is a pair $(a, b) \in (\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\})$ $\{(\lambda, \lambda)\}$. It is usually written as a $\rightarrow$ b. An *alignment* A of two strings σ_1 and σ_2 is a sequence $(a_1 \rightarrow b_1, ..., a_h \rightarrow b_h)$ of edit operations such that $\sigma_1 = a_1...a_h$ and $\sigma_2 = b_1...b_h$. A *weight function* δ assigns to each edit operation $a \rightarrow b, a \neq b$ a positive real weight $\delta(a \rightarrow b)$. The weight $\delta(a \rightarrow a)$ of an edit operation $a \rightarrow a$ is 0. If $\delta(a \rightarrow b) = 1$ for all edit operations $a \rightarrow b$, $a \neq b$, then δ is the *unit weight function.* The weight $\delta(A)$ of an alignment A is defined by $\delta(A) = \Sigma_{a \rightarrow b \in A} \delta(a \rightarrow b)$. The *weighted edit distance* of σ_1 and σ_2 is the minimum possible weight of an alignment of σ_1 and σ_2 [3].

3.2 Q-Grams

Given a string σ, its *q-grams* are obtained by "sliding" a window of length q over the characters of σ. Since *q-grams* at the beginning and the end of the string can have fewer than q characters from σ, I introduce a new character "#" which is *not* in Σ, and conceptually extend the string σ by prefixing and suffixing it with $q-1$ occurrences of "#". Thus, each *q-gram* contains exactly q characters, though some of these may not be from the alphabet Σ [4]. G_σ denotes the list of all the $|\sigma| + q - 1$ q-grams of σ.

The intuition behind the use of q-grams as a foundation for approximate string processing is that when two strings σ_1 and σ_2 are within a small edit distance, they share a large number of q-grams. The q-gram distance qgram(σ_1, σ_2) between two strings σ_1 and σ_2 is defined as follows: [3]

$$qgram(\sigma_1, \sigma_2) = \frac{1}{2}(\frac{|G_{\sigma_1} \cap \sigma_2|}{|G_{\sigma_1}|} + \frac{|G_{\sigma_1} \cap \sigma_2|}{|G_{\sigma_2}|}) \tag{1}$$

4 Approximate Matching of Data Records from Different Databases

This section describes in detail the various steps followed to match the data at the Municipality. Figure 2 illustrates various steps followed to match 3 autonomous heterogeneous databases of various departments at the Municipality of Bolzano-Bozen. The work is divided into four main parts:

1. Cadastre Data Extraction
2. Import of the Text Files
3. Data Cleaning
4. Data Matching

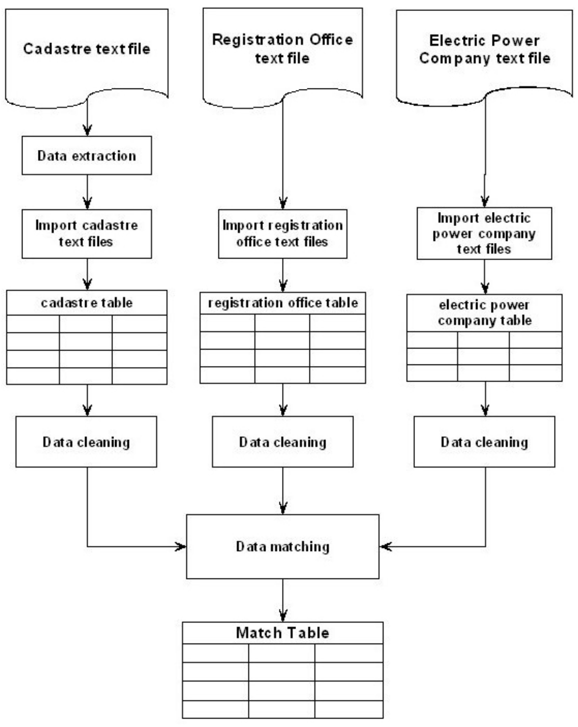
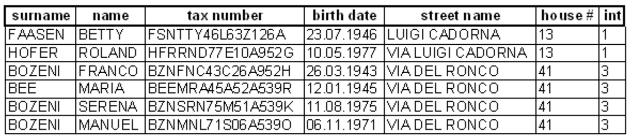

Cadastre text file
Registration Office text file
Electric Power Company text file
Data extraction
Import cadastre text files
Import registration office text files
Import electric power company text files
cadastre table
registration office table
electric power company table
Data cleaning
Data cleaning
Data cleaning
Data matching
Match Table

Figure 2: Flowchart diagram

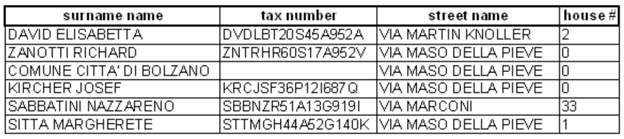
4.1 Source Data Files

The starting point are the text files provided by different departments. Each single department provided a text file, semicolon or tab separated, of his database respectively. A description of the provided data



surname	name	tax number	birth date	street name	house #	int
FAASEN	BETTY	FSNTTY46L63Z126A	23.07.1946	LUIGI CADORNA	13	1
HOFER	ROLAND	HFRRND77E10A952G	10.05.1977	VIA LUIGI CADORNA	13	1
BOZENI	FRANCO	BZNFNC43C26A952H	26.03.1943	VIA DEL RONCO	41	3
BEE	MARIA	BEEMRA45A52A539R	12.01.1945	VIA DEL RONCO	41	3
BOZENI	SERENA	BZNSRN75M51A539K	11.08.1975	VIA DEL RONCO	41	3
BOZENI	MANUEL	BZNMNL71S06A539O	06.11.1971	VIA DEL RONCO	41	3

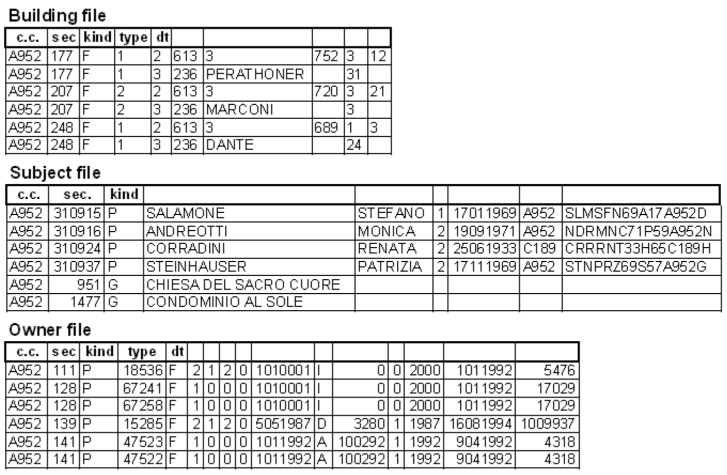
Figure 3: Semicolon separated registration office text file



surname name	tax number	street name	house #
DAVID ELISABETTA	DVDLBT20S45A952A	VIA MARTIN KNOLLER	2
ZANOTTI RICHARD	ZNTRHR60S17A952V	VIA MASO DELLA PIEVE	0
COMUNE CITTA' DI BOLZANO		VIA MASO DELLA PIEVE	0
KIRCHER JOSEF	KRCJSF36P12I687Q	VIA MASO DELLA PIEVE	0
SABBATINI NAZZARENO	SBBNZR51A13G919I	VIA MARCONI	33
SITTA MARGHERETE	STTMGH44A52G140K	VIA MASO DELLA PIEVE	1

Figure 4: Semicolon separated Electric Power Company text file

is deliberate together with the data.
The *registration office text file* stores personal data (*surname, name, birth date and tax number*) of citizens which are registered and live in Bolzano-Bozen and their primary address (*street name, house number, internal apartment number...*) (Figure 3 on page 6).
The *electric power company text file* is a semicolon separated file that stores personal data (*surname name as joined information and tax number*) and addresses (*street name, house number, internal apartment and floor number*) for the citizens that pay the electricity bill (see Figure 4 on page 6). Every person who wants electric power in his department has also to pay the bills and therefore it is registered at the electric power company.The residence data is accurate because of the periodically electricity meter inspection of an employee of this department. The data regarding the person is less accurate.
The *cadastre text file* provides 3 different files. These three tab separated files are generated from their hierarchical database holding information about the apartments (*street name, house number, internal, floor, parcel of place, cost of the parcel, number of the parcel...*) and the persons (*surname, name, birth date, tax number, place of birth...*) owing the apartments in the city (Figure 5 on page 7). The cadastre is the department that controls all the real properties and their owners of the municipality of Bolzano-Bozen. It holds personal data as well as detailed information about the properties located in the Municipality of Bolzano-Bozen.



Building file

c.c.	sec	kind	type	dt					
A952	177	F	1	2	613	3	752	3	12
A952	177	F	1	3	236	PERATHONER		31	
A952	207	F	2	2	613	3	720	3	21
A952	207	F	2	3	236	MARCONI		3	
A952	248	F	1	2	613	3	689	1	3
A952	248	F	1	3	236	DANTE		24	

Subject file

c.c.	sec.	kind						
A952	310915	P	SALAMONE	STEFANO	1	17011969	A952	SLMSFN69A17A952D
A952	310916	P	ANDREOTTI	MONICA	2	19091971	A952	NDRMNC71P59A952N
A952	310924	P	CORRADINI	RENATA	2	25061933	C189	CRRRNT33H65C189H
A952	310937	P	STEINHAUSER	PATRIZIA	2	17111969	A952	STNPRZ69S57A952G
A952	951	G	CHIESA DEL SACRO CUORE					
A952	1477	G	CONDOMINIO AL SOLE					

Owner file

c.c.	sec	kind	type	dt											
A952	111	P	18536	F	2	1	2	0	1010001	I	0	0	2000	1011992	5476
A952	128	P	67241	F	1	0	0	0	1010001	I	0	0	2000	1011992	17029
A952	128	P	67258	F	1	0	0	0	1010001	I	0	0	2000	1011992	17029
A952	139	P	15285	F	2	1	2	0	5051987	D	3280	1	1987	16081994	1009937
A952	141	P	47523	F	1	0	0	0	1011992	A	100292	1	1992	9041992	4318
A952	141	P	47522	F	1	0	0	0	1011992	A	100292	1	1992	9041992	4318

Figure 5: The three tab separated cadastre text files

4.2 Cadastre Data Extraction

The data provided by the cadastre is not suitable for a relational database. The cadastre department provides a zip file that contains 3 tab separated files. The 3 cadastral text files (see figure 5 on page 7) have to be parsed to extract the required information.

The first input file (see figure 5), the *Building File* contains information about the building. The first 6 attributes of each record form the key of the record. One of these attributes contain a number that determines the type of data. With respect to this number we can parse the text file and there are generated five new output files containing structured information. The generated text files are the following: *1) real estate, 2) identification, 3) address, 4) municipality, 5) reserve.* Each of these files contain different information about an object. A record has not a fixed length of data attributes, which means that data of 2 records can be merged in one single record. If this is the case the record is split in 2 single records. A second input file, the *Subject File*, contains information about the subject or person. The first 4 attributes of each record form the key of the record. One of these attributes contain a character that determines the type of data (P stands for physical person, G stands for juridical person). With respect to this character we can parse the text file and two new output files containing structured information are generated. The new generated files are: *1) physical person, 2) legal person.* This file has not merged records on a single key which facilitates the work of splitting the data records. The third file, the *Owner File*, with information

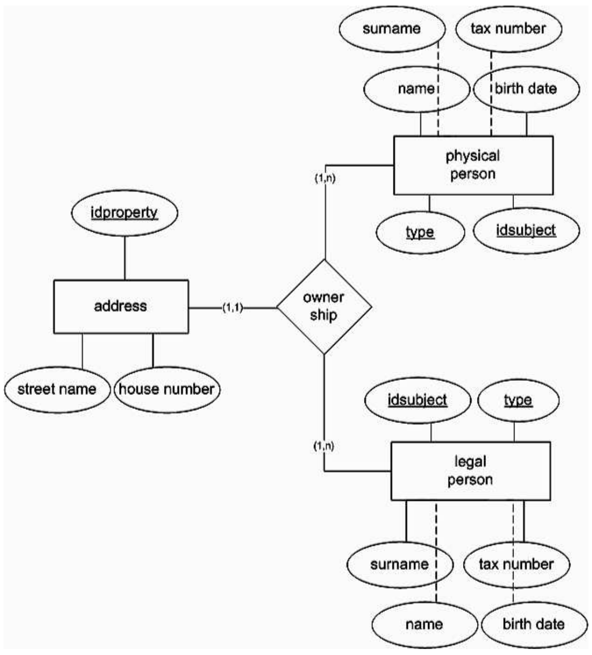


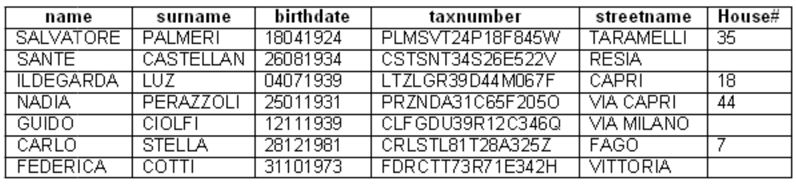
Figure 6: Entity relationship diagram of the cadastre files

about the owner is already structured, because it contains data of only one type. The first 6 attributes of each record form again the key of the record.

The data provided as text files by the cadastre department, have been extracted out of their hierarchical database. As we want to import the data in a relational database this data is not suitable in the form it is provided by the cadastre. We have to bring it on a structure of a relational database for a correct import in a MySql DBMS (Database Management System). By studying the content of the files and applying the description of the data from the "data set structure manual" provided by the cadastre, we are able to build an ER-diagram (see figure 6 on page 8). The ER-diagram shows the relationship between the tables and which attributes we find in each relation.

4.3 Import Text Files

This step requires the import of the text files in a database. The following three imports take place: *import cadastre text files*, *import registration office text file*, *import electric power company text file*. In our project we use the MySql DBMS. The data we want to work with



name	surname	birthdate	taxnumber	streetname	House#
SALVATORE	PALMERI	18041924	PLMSVT24P18F845W	TARAMELLI	35
SANTE	CASTELLAN	26081934	CSTSNT34S26E522V	RESIA	
ILDEGARDA	LUZ	04071939	LTZLGR39D44M067F	CAPRI	18
NADIA	PERAZZOLI	25011931	PRZNDA31C65F205O	VIA CAPRI	44
GUIDO	CIOLFI	12111939	CLFGDU39R12C346Q	VIA MILANO	
CARLO	STELLA	28121981	CRLSTL81T28A325Z	FAGO	7
FEDERICA	COTTI	31101973	FDRCTT73R71E342H	VITTORIA	

Figure 7: Joined data of the cadastre files

has to be imported in a database. Each table represents a department of the Municipality of Bolzano-Bozen. We import all the text files in a database. The registration office and the electric power company files, which are provided as *.csv files, have already present all their information in one text files. The tab separated cadastre files are imported in the database. Following the ER-diagram (see figure 6 on page 8) we join the data and collect the needed data in one single table, to be compatible with our matching algorithms (see figure 7 on page 9). and the tables are joined.

4.4 Data Cleaning

The data present in the tables is heterogeneous and shows different levels of accuracy. The fact that many persons insert records and administrate the databases creates a variation in the data. In order to achieve a good matching level the data containing errors has to be cleaned in advance. Data with well known pattern has to follow the standard defined rules. Following this basic rules we will be able to achieve a better similarity between the matching records. The cleaning algorithms are applied on the three tables used for the matching. The cleaning algorithms concentrate on the six data fields which are used to build the matches. This cleaning procedure produces a better approach by executing simply a MySql query and also for the developed matching algorithms. With the cleaning procedure it is also intended to remove its relevant data, which means columns not present in all three databases are deleted because they are not relevant for the matching algorithms.

The tables demonstrate a different level of irregularity. The cadastre table has many irregularities in it. Because of the absence of standardized rules for the data insertions and since its digital birth in the latest '70 the data became more and more heterogenous.

The electrical power company provides a table holding old street names. Through a translation table holding the registration office and the electrical power company the old street names are updated with the newer once. This update affects 120 of the total 301 streets. 18339 records are updated. This correction is done using standard MySql updates.

4.4.1 Checked Attributes

The *surname* as well as the *name* of a person is scanned for non valid characters. A person can only contain characters and spaces. Numbers and special characters are recognized as suspicious data (e.g. Marcheggiani Claudi0, Pez7o Giuseppe).
The *tax number* of a physical person is checked for its well defined pattern. The first 6 are characters which identify the surname and name of the person followed by two numbers specifying the year, followed by a character for the month and two numbers regarding the day of birth. After this a character and 3 numbers specify the place of birth concluding with a check character bit. Entries having a different data pattern are marked as suspicious data.
The *street name* is checked for non legal characters such as numbers and special characters (Via Maso De11a Pieve, Via R@ma).
The *house number* is checked for non legal characters. Slashes can indicate a building that contain two house numbers, which is the case when a house borders on two streets (e.g. 32/34). In this case the 2 house numbers could be separated for generating a new record. If a character is found near a house number (e.g. 2D) it will probably be a division number for this building. In this case the character could be a division attribute. The *birth date* of a person is checked for its "day.month.year" (dd.mm.yyyy) pattern, with no characters spaces or other not valid characters in it. The similar (ddmmyyy) pattern with missing punctation is also a valid pattern because it is seen as a special case in the matching algorithm.

4.4.2 Error Checking

The suspicious data is find using *regular expressions*. A regular expression (sometimes abbreviated to "regex") is a way for a computer user or programmer to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found.

Example: *The regex* `.{6}\d{2}.{1}\d{2}.{1}\d{3}.{1}` *checks the correctness of the* tax number *like* `(SCHNFR74T23A952R)`. *The regular expression specifies the format of a standard tax number which is: 6 characters, 2 digits, 1 character, 2 digits, 1 character, 3 digits and 1 character. Example of false tax numbers found* `(LTRPTR23E21E959,`

PRFDSG12T23984D).
The regex \d+ checks the exactness of names. *No digits are allowed in this pattern (e.g. Zona 1ndustraile, Mari0, Costan7i).*
The regex \d{2}.?\d{2}.?\d{4} controls the format of the birthdate which has to be of type (dd.mm.yyyy or ddmmyyy). Non admissible dates are found e.g. 21.111980, 3.09.1979

4.5 Data Matching

When the data is cleaned it is ready to be matched. The matching algorithm requires 2 relations with residential and personal data used as matching attributes. For each entry in one relation we want to find the best match(es) in the other relation.

4.5.1 Similarity Measure

The similarity between two tuples is calculated using some string matching techniques and true/false checking. The string matching algorithms used in our project are: edit distance and q-gram algorithm described in section 3. The distance threshold of two matching records is expressed in a range from 0 (best case) to 100 (worst case). The distance d between two tuples t_1 and t_2 is calculated by summing up the distance of the matching attributes (street name, house number, name, surname, tax number and birth date) multiplied by their assigned weight:

$$d(t_1, t_2) = \sum d(a_j, b_j) * w_j \qquad (2)$$

The weights have been choosen empirically. The weight is the importance of this information for finding correspondent addresses. It is not easy to find the weights for the attributes, as they can only be approximated. I have adjusted the weights by evaluating testing data. The weights are set where the best matches for the testing data are found. For the single attributes the following algorithms, multiplied by the weights are used to calculate the distances:
Edit distance algorithm is used to calculate distances where the data has to follow a defined pattern. It tells how many character changes have to be performed to get a correspondence. The algorithm is slow compared to the others used in our execution and it is used for the:

- Tax number with a weight of 23% on the similarity.

Q-gram algorithm is used to determine the distance between data with no standard pattern or given formats. It gives a distance measure based on the comparison of the tokens of the compared attributes. It is fast in its execution and used for:

- The surname weighting 15% on the similarity
- The name weighting 15% on the similarity
- The street name weighting 23% on the similarity

True/False statement is used to check if two pieces of data are equal. It is very fast and used for:

- The birthdate is splitted up into 3 tokens which are the day, the month and the year. The two records match if the three tokens match exactly. The three tokens are checked with an equal not equal procedure weighting 15% on the similarity.
- The house number is checked if it is equal or not equal weighting 8% on the similarity.

Example: *Distance measure of two tuples. The distance is calculated for each attribute multiplied by its weight. The sum of the distances gives the total distance measure. The distance of the name, surname and street name are calculated using the Q-gram algorithm. The edit distance algorithm is applied for the tax number. The birth date and the house number are checked with true/false statements.*

tuple 1	Verdi	Stefano	23.12.1943	PRZSTF43T28A952F	Via Fago	6	
distance	0.0 +	5.4 +	0.0 +	0.0 +	8.2 +	0.0 +	= 13.6
weight	15%	15%	15%	23%	23%	8%	
tuple 2	Verdi	Stefan	23121943	PRZSTF43T28A952F	Fago	6	

4.5.2 Matching Records on Personal and Address Data

Two records are said to be equal if their personal data and address are matching. For personal data the surname, name, tax number, birth date are the matching columns. The address has the street name and the house number as matching columns. Joining the tables with standard join techniques (SQL-join) ends up with very poor results, as only for about 2% of the records an exact matches is found. The problem hereby is, that some databases do not have a common key, that could be used as a join attribute. The most common attributes found in the tables of the databases are the personal and residential data. So, to get virtually one database, there has to be performed a join on this attributes. Approximate string matching techniques, by calculating similarities between the records of the tables, produce a much better approach. Figure 8 on page 13 shows the pseudo code of the algorithm that calculates the similarities. It takes 2 relations and a distance threshold as input. The algorithm searches for each record in $table_1$ the best (can be one ore more) records in $table_2$. The similarity of a record is calculated by summing each single similarity

```
function calculateSimilarity($R_1$, $R_2$, d)
// Input:  $R_1$ and $R_2$, data of two tables
// Output:  a relation containing the matching records
$d$:  distance threshold
$min$:  minimum achieved distance
$S$:  empty stack of tuples($t_1, t_2$)
M($t_1, t_2, dist$):  empty matching relation
for each tuple $t_1$ in $R_1$
    for each tuple $t_2$ in $R_2$ do
        $dist$:=sim($t_1$.taxnumber, $t_2$.taxnumber) * tax-weight +
                sim($t_1$.streetname, $t_2$.streetname) * street-weight +
                sim($t_1$.housenumber, $t_2$.housenumber) * house-weight +
                sim($t_1$.surname, $t_2$.surname) * surname-weight +
                sim($t_1$.name, $t_2$.name) * name-weight +
                sim($t_1$.birthdate, $t_2$.birthdate) * birth-weight +
        if $dist <= d$
            if $dist < min$
                $min = dist$
                $S$.empty()
                $S$.push(($t_1, t_2, dist$))
            else
                $S$.push(($t_1, t_2, dist$))
            endif
        endif
    enfor
    $M = M \cup S$.popAll()
endfor
return M
```

Figure 8: Algorithm that calculates the similarities of 2 tables

(tax number, street name, house number, surname, name and birthdate) multiplied by the assigned weight. The matched records below or equal the distance threshold limit are put in a stack. The stack holds only matches which comprising the same similarity value. If a better match is found, the old stack elements are deleted and the better match is inserted. The found matches and their similarities are inserted in a matching table called M. At the end the algorithm returns the matching relation.

4.5.3 Matching Table

The matching between the 2 tables is done by checking common attributes contained in both tables. The matching table contains three main parameters that can be used to extract all other data fields present in the original relations in a second moment. The parameters are two key values of the records and the similarity between them (see table 1 on page 14). The similarity is the sum of all deviations of the specified matching attributes e.g. surname, name, birth date, tax

MATCHING T

id1	sim	id2
5	4.3	277
84	8.7	732
174	13.2	928
732	6.3	43
865	11	153
932	3.2	635
1043	12.4	234
...	...	...

TABLE 1

id	tax #	name	street	#	...
5	smg934s	Daniel	Viale Druso	3	...
84	rft983k	Stefan	Via Fago	23	...
174	rpc084f	Anna	Via Roma	5	...
732	hdm327j	Mara	Corso Italia	19	...
...	...	...	...	...	...

TABLE 2

id	tax #	name	street	#	...
43	hdm327j	Mara	Italia	19	...
277	smg934s	Daniel	Druso	3	...
732	rft983k	Stefan	Fago	23	...
928	rpc084f	Anna	Roma	5	...
...	...	...	...	...	...

Table 1: Matching Table example: correspondences in 2 tables

number, street name and house number. The matching table is saved as new relation in the database, so once created, it is possible to access and extract relevant data of the matching records.

4.5.4 Improving Efficiency

Street names reveal many irregularities, this means a street name can be written in different forms *(e.g. Leonardo Da Vinci, L. da Vinci, Leonardo da V., L.D. Vinci)* because of the spelling mistakes the different insertion users made. Sometimes these are not mistakes, the street names can be abbreviated or just reduced in a time-spare spelling form. This not standardized way of inserting the data causes inconsistency during the years.

In the case of the registration office and electric power company, the street names are consistent, which means that one street has exactly one possible spelling *(e.g. Corso Della Libertà).* Beside the cadastre data file has many irregularities in the street names, which means that one street is written in different ways, or spelling mistakes were made *(e.g. Corso Della Libertà, C.D. Libertà, Corso Della Liberta, Corso Libertà).* In the registration office and the electric power company we can find 304 different street names which is in fact the complete number of streets we can find in the territorial of the Municipality of Bolzano-Bozen. The amount of streets in the cadastre is equal to 1102, which is equivalent to 3,7 different spellings.

When we have to create the matchings between the two tables the check of the street similarity is very time consuming. Taking in consideration that each record in one data file has to be compared

Key	**Similarity value**							
1	0.13	0.201	0.14	0.432	0.012	0.032	0.23	...
2	0.230	0.02	0.201	0.022	0.201	0.310	0.012	...
3	0.21	0.072	0.391	0.032	0.324	0.231	0.105	...
4	0.402	0.021	0.101	0.052	0.004	0.023	0.210	...
5	0.01	0.103	0.031	1.0	0.102	0.074	0.105	...
6	0.056	0.082	0.002	0.402	0.205	0.032	0.099	...
7	0.063	0.084	0.092	0.063	0.153	0.32	0.032	...
8	0.023	0.043	0.320	0.053	0.092	0.207	0.085	...
...	...	...	...	...	...	...	...	...
Key	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**...**

Table 2: Similarity matrix: table1/table2 street similarities

TABLE 1

Key	**Value**
3 Santi	1
A. Adige	2
Corso Liberta	20
Della Mendola	83
Lungo Adige	287
P.E.Eugenio Di Savoia	307
Passegg. S. Osvaldo	371

TABLE 2

Key	**Value**
Corso Della Libertà	6
Galleria Vintler	32
Lungo Adige	94
Piazza Del Grano	132
Via Alto Adige	203
Via Bassano Del Grappa	213
Via Tre Santi	224

Table 3: Hash-table maps keys to the streets

with 100.000 records in the other data file. String matching algorithms would require several days of work which is not suitable for our matching algorithm. A better solution has to be found.

By constructing an m*n matrix containing the similarities of all the street names the similarities have to be calculated only once. Table 2 on page 15 shows the matrix, where each street name is represented by a key value and the corresponding similarities.

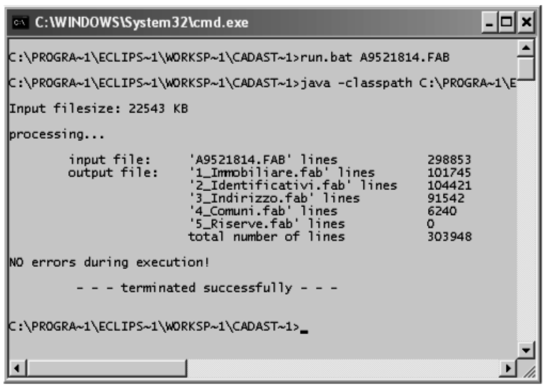
Each street gets a key in the matrix. This keys are mapped through a hash-table which maps keys to values. Such mapping is shown in table 3. Each street name gets an integer number which serves as key in the similarity matrix. The similarities for each street are calculated only once at the beginning of the execution of the program. Each time a similarity is needed, the keys of the streets are extracted from the hash-table and with this keys the matrix can be indexed on the right position and the similarity can be extracted without any calculation.

5 Implementation

This subsection demonstrates the procedures followed in the realization of this project, the main functionalities and details. The implementation of the project required the creation of two standalone applications. The first one is a command line tool which parses the data out a text file. The second application takes the data imported in a database system, detects suspicious data and creates a matching between three tables.

5.1 Cadastre Data Wrapping

The "CADASTRE DATA WRAPPING" tool (figure 9 on page 16) is a program which processes the files (*.fab, *.sog, *.tit) downloadable from the cadastre department of the provence of Bolzano and Trento "www.catastobz.it". This program generates new files following the structure of this data which is specified on the same site using regular expressions.



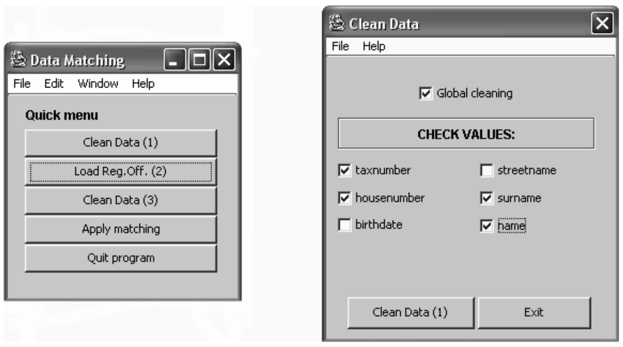
```
C:\WINDOWS\System32\cmd.exe

C:\PROGRA~1\ECLIPS~1\WORKSP~1\CADAST~1>run.bat A9521814.FAB

C:\PROGRA~1\ECLIPS~1\WORKSP~1\CADAST~1>java -classpath C:\PROGRA~1\E

Input filesize: 22543 KB

processing...

        input file:     'A9521814.FAB' lines            298853
        output file:    '1_Immobiliare.fab' lines       101745
                        '2_Identificativi.fab' lines    104421
                        '3_Indirizzo.fab' lines         91542
                        '4_Comuni.fab' lines            6240
                        '5_Riserve.fab' lines           0
                        total number of lines           303948

NO errors during execution!

         - - - terminated successfully - - -


C:\PROGRA~1\ECLIPS~1\WORKSP~1\CADAST~1>_
```

Figure 9: Cadastre Data Wrapping tool

Start the program by typing run.bat and after a white space specify the path and the name of the input file along with the file extension. If the input file is in the same directory as the start file just specify the name of the input file without path. e.g. `(run.bat A9521814.fab)` If the input files contain data that can not be handled, an error file named "error.txt" is generated. It lists all lines which contain an error an therefore could not be handled by the CADASTRE DATA WRAPPING tool. To correct the errors just open the input file and adjust the specified lines. This tool produces a file with the errors and additional 5 output files containing the data of 5 different types for the input file **.fab: 1) Immobiliare.fab, 2) Identificativi.fab, 3) Indirizzo.fab, 4) Comuni.fab, 5) Riserve.fab* and 2 output files containing the data of 2 different types for the input file **.sog: 1) Persona fisica.sog, 2) Persona giuridica.sog.* The input file **.tit* is already in an import compatible format. It is scanned for bad line format.

5.2 Data Matching

The main application is the cadastre, registration office and electrical power company matching tool. It reads data from a MySql database, cleans it and searches matches in all three tables. The first step is to load the data from the MySql database. Three buttons for the loading of the data are placed in the start panel (see figure 10). You have to load the tables by specifying some basic loading parameters, such as database-name, host and port, table name, column names, username and password. After the data is loaded the button changes its text to "clean data (x)". The button can be pushed again to specify the parameters for the data cleaning (see figure 10).



✓ Global cleaning

✓ taxnumber ☐ streetname
✓ housenumber ✓ surname
☐ birthdate ✓ name

Figure 10: "Data Matching" GUI and "Clean Data" dialog

The Clean Data dialog contains checkboxes that allow the specification of the columns that should be controlled and checked. Basic errors like a whitespace is directly changed in the MySql table by the programm. Other suspicious data regarding surname, name, tax number, birth date, street name or house number is written out in a text file. The user can open the file and through the identification number and the fault of each record it is possible to correct the data manually.

Example error log file:

```
suspected tax number in table cadastre
    387 LTRPTR23E21E959
```

```
1043 FRFHDC12TE5T952D
- - - END - - - number of rows 2
```

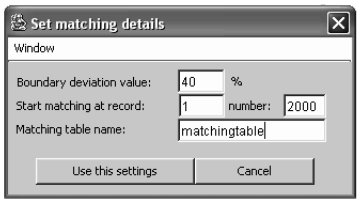
suspected house number in table table cadastre

```
8231 51@53
8679 51F
39425 7.
74099 4@12
- - - END - - - number of rows 4
```

suspected birthdate in table table cadastre

```
3987 2105192
6234 30.5.1947
- - - END - - - number of rows 2
```

When the data is cleaned it is ready to be matched. When the button "apply matching" is pushed a new dialog appears. The boundary deviation value specifies which similarity limit we want to apply for the matching, 0 specifies the best matching value, 100 is the worst case. We can also input the value at which record the matching should start and at which record it should stop (see figure 11 on page 18). This can be an advantage, because on old or slow computers the calculation can take much time. After the specification of this values the matching starts.



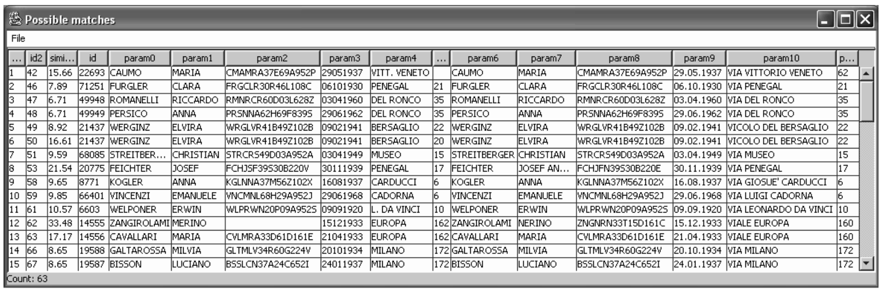
Boundary deviation value: 40 %
Start matching at record: 1 number: 2000
Matching table name: matchingtable

Figure 11: Dialog for specifying the matching details

The tool tries to find for the first record the best matching record in the registration office. The similarity is calculated on two decimals exactness. If more than 1 record was found for one record in the cadastre the results are saved in an array and recognized as matching records. The generated matching table inserts more new matching records. For the matching records on the registration office side correspondences are searched in the electrical power company table. The matching parameters for the same matching algorithms are the tax number, street name and house number. Only this three fields can be used because they are the only common columns for the two tables.

A status line informs the user about the records that come up and the similarity values.

The values are written in a MySql table for further usage, while the user can see the results by pushing the "Show Matches button" in the main GUI (Graphical User Interface). A table containing all the matching details and parameters give the user the opportunity to check the obtained results just after the matching procedure (see figure 12 on page 19). The table shows also the matching data (surname, name, street name...) for a better visualization of the matches.



...	id2	simi...	id	param0	param1	param2	param3	param4	...	param6	param7	param8	param9	param10	p...
1	42	15.66	22693	CAUMO	MARIA	CMAMRA37E69A952P	29051937	VITT. VENETO		CAUMO	MARIA	CMAMRA37E69A952P	29.05.1937	VIA VITTORIO VENETO	62
2	46	7.89	71251	FURGLER	CLARA	FRGCLR30R46L108C	06101930	PENEGAL	21	FURGLER	CLARA	FRGCLR30R46L108C	06.10.1930	VIA PENEGAL	21
3	47	6.71	49948	ROMANELLI	RICCARDO	RMNRCR60D03L628Z	03041960	DEL RONCO	35	ROMANELLI	RICCARDO	RMNRCR60D03L628Z	03.04.1960	VIA DEL RONCO	35
4	48	6.71	49949	PERSICO	ANNA	PRSNNA62H69F839S	29061962	DEL RONCO	35	PERSICO	ANNA	PRSNNA62H69F839S	29.06.1962	VIA DEL RONCO	35
5	49	8.92	21437	WERGINZ	ELVIRA	WRGLVR41B49Z102B	09021941	BERSAGLIO	22	WERGINZ	ELVIRA	WRGLVR41B49Z102B	09.02.1941	VICOLO DEL BERSAGLIO	22
6	50	16.61	21437	WERGINZ	ELVIRA	WRGLVR41B49Z102B	09021941	BERSAGLIO	20	WERGINZ	ELVIRA	WRGLVR41B49Z102B	09.02.1941	VICOLO DEL BERSAGLIO	22
7	51	9.59	68085	STREITBER...	CHRISTIAN	STRCRS49D03A952A	03041949	MUSEO	15	STREITBERGER	CHRISTIAN	STRCRS49D03A952A	03.04.1949	VIA MUSEO	15
8	53	21.54	20775	FEICHTER	JOSEF	FCHJSF39S30B220V	30111939	PENEGAL	17	FEICHTER	JOSEF AN...	FCHJFN39S30B220E	30.11.1939	VIA PENEGAL	17
9	58	9.65	8771	KOGLER	ANNA	KGLNNA37M56Z102X	16081937	CARDUCCI	6	KOGLER	ANNA	KGLNNA37M56Z102X	16.08.1937	VIA GIOSUE' CARDUCCI	6
10	59	9.85	66401	VINCENZI	EMANUELE	VNCMNL68H29A952J	29061968	CADORNA	6	VINCENZI	EMANUELE	VNCMNL68H29A952J	29.06.1968	VIA LUIGI CADORNA	6
11	61	10.57	6603	WELPONER	ERWIN	WLPRWN20P09A952S	09091920	L. DA VINCI	10	WELPONER	ERWIN	WLPRWN20P09A952S	09.09.1920	VIA LEONARDO DA VINCI	10
12	62	33.48	14555	ZANGIROLAMI	MERINO		15121933	EUROPA	162	ZANGIROLAMI	NERINO	ZNGNRN33T15D161C	15.12.1933	VIALE EUROPA	160
13	63	17.17	14556	CAVALLARI	MARIA	CVLMRA33D61D161E	21041933	EUROPA	162	CAVALLARI	MARIA	CVLMRA33D61D161E	21.04.1933	VIALE EUROPA	160
14	66	8.65	19588	GALTAROSSA	MILVIA	GLTMLV34R60G224V	20101934	MILANO	172	GALTAROSSA	MILVIA	GLTMLV34R60G224V	20.10.1934	VIA MILANO	172
15	67	8.65	19587	BISSON	LUCIANO	BSSLCN37A24C652I	24011937	MILANO	172	BISSON	LUCIANO	BSSLCN37A24C652I	24.01.1937	VIA MILANO	172

Count: 63

Figure 12: Possible matches produced by the application

5.3 Java Implementation

The applications developed for this project are java applications. For writing the code I used the Eclipse Platform. We have choosen these technologies, because they are open source and run on every machine. The first application which is the cadastre data wrapping tool is a command line tool an it contains 2 classes and about 300 lines of code. A code example that splits the file depending on the content of each line using regular expressions is shown in figure 13 on page 20. The second tool, the data matching tool is a java application. This tool creates a connection to a MySql database, using the jdbc drivers. This tool contains 18 classes which are structured in 6 packages and contain about 4000 lines of code. The java code example of the matching algorithm is shown in figure 14 on page 21. We use the MySql version 4.1 adapted to support nested queries, that where very useful to get an insight on the data received by the departments of the Municipality and to get a better overview of the domain.

```
//splits the lines of type PROPERTY (*.fab)
public void arcImmobili()throws IOException{
DataExtraction de = new DataExtraction("immobili");
     int lineNum = 0;
     String matchLine;
     String standardPat = "A952//|//|[0 − 9]{1,9}//|F//|[0 − 9]{1,3}//|";
     for (int u = 1; u < 6; u + +) {
          pat[u] = Pattern.compile(standardPat + u + "//|.*");
     }
     while ((matchLine = in.readLine()) != null) {
          lineNum + +;
          for (int u = 1; u < 6; u + +) {
               mat[u] = pat[u].matcher(matchLine);
          }
          if (mat[1].matches()) {
               de.landedProperty(matchLine, count);
          } else if (mat[2].matches()) {
               de.keyData(matchLine, count);
          } else if (mat[3].matches()) {
               de.address(matchLine, count);
          } else if (mat[4].matches()) {
               de.combGoods(matchLine, count);
          } else if (mat[5].matches()) {
               de.landedProperty(matchLine, count);
          } else {
               de.writeError(("unresline" + lineNum + matchLine));
               DataExtraction.isError = true;
               DataExtraction.numErrors++;
          }
     }
}
```

Figure 13: Java code that splits a file using regular expressions

```
//limit:  distance threshold
//rs1, rs2:  the MySql reslutset of to relations
public void computeSimilarity(float limit, Resultset rs1, Resultset rs2){
    while(rs1.next()){
        while(rs2.next()){
            //compute similarity for the 6 matching attributes
            for(int i = 0; i < 6; i++){
                if(rs1.getString(i) == ""rs2.getString(i) == "")
                    sim=0;
                else{
                    //distance algorithms
                    sim =+ editDist(rs2.getString(i),rs1.getString(i))*Weight;
                    sim =+ q-gram(rs2.getString(i),rs1.getString(i))*Weight;
                    sim =+ true/false(rs2.getString(i),rs1.getString(i))*Weight;
                    totalWeight = +simWeight;
                }
            }
            sim = /totalWeight;
            if(sim <= reachedSim){
                if(reachedSim == sim){
                    equalSim++;
                    eSe[equalSim] = new double[]{rs2.getString(id2),
                    sim, rs1.getString(id1)};
                }
                else{
                    equalSim=0;
                    eSe[equalSim] = new double[]{rs2.getString(id2),
                    sim, rs1.getString(id1)};
                }
                reachedSim = sim;
            }
        }
        if(reachedSim <= limit){
            for(int z = 0; z <= equalSim; z++){
                //only elements with lowest similarity
                if(eSe[z][8] == reachedSim)
                    insertTable("insert into match table values +
                    ('"+eSe[z][1]+"','"+eSe[z][1]+"','"+eSe[z][2]+"');");
            }
        }
        rs2.first();
    }
}
```

Figure 14: Java code of the matching algorithm

6 Evaluation

In this subsection I evaluate the final result founded by applying different string matching algorithms on the records of the different tables. The similarity of two matched records has a range from [0 to 100], whereas 0 is the best case said two records match exact and 100 is the worst case where each of the matching attributes has no similarity.

6.1 Match Cadastre - Registration Office Data

We want to find the correspondent matches in our cadastre relation (112.000 insertions) and the registration office (97.000 insertions). The application takes about 50 hours ($\approx$ 2 days) to calculate the correspondences in the tables. For each record in the first table the similarity for all records in the second table have to be calculated (112.000*97.000). Figure 15 on page 23 shows the distribution of the similarities of the matching records. The approximate matching is done using personal data (surname, name, birth date and tax number) and residential data (street name and house number). This data fields are present in both of the matched relations. It shows the matches found in the registration office relation for 5000 records of the cadastre relation. The experiment shows an irregular distribution of the similarities.

[**0-10**] 34% of the matches

[**10-20**] 23% of the matches

[**20-30**] 10% of the matches

[**30-100**] 33% of the matches

This is due to the fact that the relations hold many data with many irregularities. The algorithm is able to find about 2% of exact matches. This means that about 2000 records have a 1:1 correspondence. The other records are possible matches and have to be evaluated. The peak reached nearby the deviation value equal to 10 is very probable to be a match with some irregularity in the street name. The registration office relation holds only the 304 valid street names in the Municipality of Bolzano-Bozen *(e.g. Viale Druso, Via Maso Della Pieve...)*, where the cadastre follows no rules by inserting the street names *(e.g. Viale Druso, V. Druso, Via M.D.Pieve, V. Maso Della Pieve, Via Maso D.Pieve...)*. The big deviation of the matches is due to the 1103 street names of the cadastre compared with the 304 of the registration office. An other common error is the spelling of the names *(e.g. Stefan-Stefano, Karl-Carlo...)*.

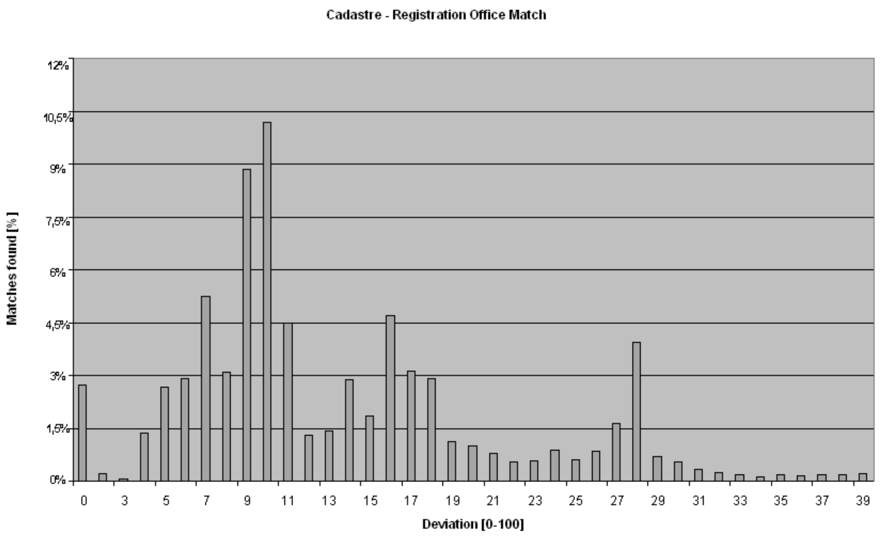


Figure 15: Cadastre - Registration Office similarities

6.2 Match Registration Office - Electric Power Company Data

We want to find the correspondent matches in our registration office relation (97.000 insertions) and the electric power company (50.000 insertions). The application takes about 30 hours ($\approx$ 1,25 days) to calculate the correspondences in the tables. For each record in the first table the similarity for all records in the second table have to be calculated (97.000*50.000). Figure 16 on page 24 shows the distribution of the similarities of the matching records. The approximate matching is done using personal data present in both tables which is the tax number and residential data street name and the house number. This data fields are present in both of the matched relations. The experiment shows an irregular distribution of the similarities.

[0] 42% of the matches

[1-15] 9% of the matches

[15-30] 21% of the matches

[30-40] 28% of the matches

This is due to the fact that the relations have only three common matching attributes and therefore there is no possibility to find good distance measures for the correspondent records. We find about 42% exact matches. This means that we find 40.000 correspondent records

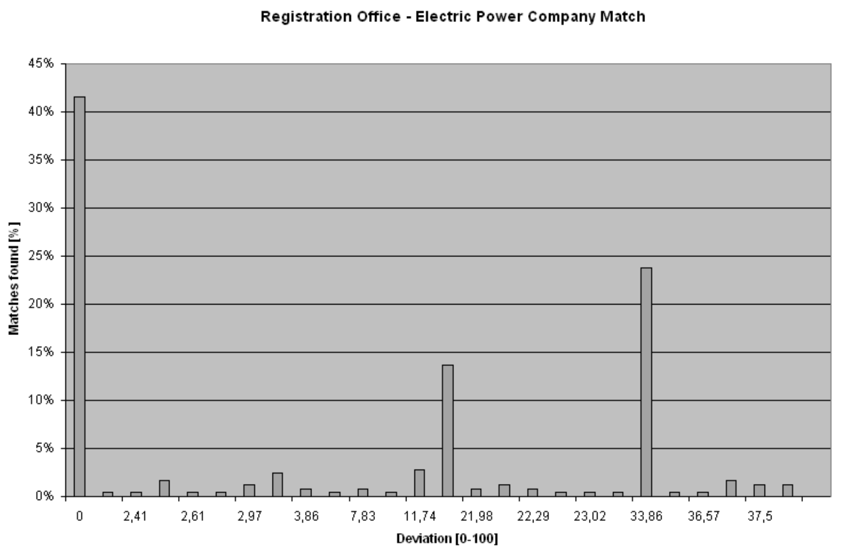


Figure 16: Registration Office - Electric Company similarities

of the total 97.000 present in the registration office database. All other records are possible matches and have to be evaluated. A peak of 14% is reached for similarity equal 19 and 24% of the matches reach a similarity of 33. This is due to the fact that some matching attributes are missed. This reached peaks arise because some records miss the tax number and therefore we can calculate the similarity only with respect to the street name and house number. By knowing only the house number and street name as residential data we are not able do find good matches.

6.3 Accuracy

Matches are found for all the records. We want to know how good the matches are and what the boundary similarity limit for correct matches is. For doing this we pick up randomly choosen matches and control their exactness. The similarity range goes from 0 to 100, where 0 are exact equal records. It is not easy to estimate the limit value for "good" matches. Sometimes a deviation value which holds a "good" match, shows no correspondence for the same deviation value. The testing on the *cadastre - registration office* matches, is done using 50 randomly choosen records. By controlling each deviation value and comparing their records, the boundary value which shows an admissible deviation of the records is 17. We are only able to estimate a value which can be seen as a good approximation for valid matches. This value is not easy to determine, because of the only manual possibility of determination. Only about 0.05% of records is taken in consideration for the testing (of 100.000 matching records we test only 50). In some cases we get a good match also with similarity value equal 30, in some cases a similarity value of 10 gives a bad match. The value 17 is a median number which was determined by the testing. 49% of the matched records have a similarity below the 17 limit. This should mean that we found 50.000 good matches for the cadastre relation, compared with the 2.000 matches found by applying standard MySql joins. The *registration office - electric power company* matches demonstrate 42% of exact matches compared with the 3% found using the standard joining techniques. All other matches have to be evaluated in detail, because only 3 matching attributes are taken in consideration.

7 Conclusion and Further Work

Different information about the citizens are used by the Municipality of Bolzano-Bozen. For some administrative tasks the data inside these heterogeneous and autonomous databases of the cadastre, registration office and the electric power company are joined. Applying standard joining techniques ends up with poor results, as there are no common keys. The data shows irregularities because of spelling mistakes, abbreviations and different naming conventions in the data fields. This work proposes a technical solution for this problem. A distance measure on common attributes of the databases, the addresses and the personal data. This measure is used to approximately join tuples coming from different databases. I have developed two tools, for applying the approximate string matching algorithms on the data. The Data Wrapping Tool is used to parse the cadastre text files and split them according to their content. The Data Matching tool is used to match the records with address and residential data. Further work for the project is a more dynamic design to support more data formats. The execution time to match the records can be improved and optimized in further projects.

Acknowledgments

I wish to thank my supervisor Prof. Johann Gamper and my second supervisor Nikolaus Augsten for supporting and coordinating my work, as well as SIT (Sistema Informativo Territoriale) at the Municipality of Bolzano-Bozen for the collaboration.

References

[1] Reducing the integration of public administration databases to approximate tree matching. Nikolaus Augsten, Michael Böhlen, and Johann Gamper. In proceedings of the Third International Conference on Electronic Government, 2004.

[2] H. V. Jagadish Nick Koudas S. Muthukrishnan Luis Gravano, Panagiotis G. Ipeirotis and Divesh Srivastava. *Approximate string joins in a database (almost) for free.* In Proceedings of the 27th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., 2001.

[3] Using approximate string matching techniques to join street names of residential addresses. Roland Innerhofer-Oberperfler. Bachelor of Science in Applied Computer Science.

[4] E. Sutinen and J. Tarhio. *On using q-gram locations in approximate string matching.* Proceedings of Third Annual European Symposium, 1995.