A NEW CYCLE OF IMPROVEMENT FOR INFORMATION QUALITY SERVICES

(Research-in-Progress)
IQ Tools

Sandra Collovini de Abreu
Acxiom Brazil, Porto Alegre, RS
sandra@acxiom.com.br

Tércio Oliveira de Almeida
Acxiom Brazil, Porto Alegre, RS
tercio.oliveira@acxiom.com.br

Luiz Carlos Ribeiro Junior
Acxiom Brazil, Porto Alegre, RS
luiz@acxiom,com.br

Marilia Terra de Mello
Acxiom Brazil, Porto Alegre, RS
marilia@acxiom.com.br

Abstract: Poor quality information produced/used by organizations impacts directly on these companies' revenues. Several losses caused by quality problems could be minimized if data quality solutions were adopted. However, due to specific needs of each organization, the process of deploying these solutions is extremely costly. Some efforts to minimize deployment costs are being made by using Supervised Machine Learning techniques. However, these approaches demand template datasets which can be a drawback to their adoption. In this paper, we present and evaluate a synthetic address data generator aiming at minimizing this problem, focusing on a new cycle of improvement for information quality services. Our proposal covers an Address Validator service, Optimizer (based on the Particle Swarm Optimization) and the data generator. The complete approach can also be generalized to other domains. In a previous work we presented the evaluation of our training process (Address Validator and Optimizer). Here we assess our synthetic address data generator, closing the proposed cycle.

Keywords: Synthetic Data Generator, Information Quality, Address Validator, Particle Swarm Optimization.

INTRODUCTION

Organizations all over the world suffer the impacts of poor quality information. Fraud, financial losses, inefficiency and poor customer relationships arise from the lack of a suitable solution to correct and prevent inconsistent, incomplete, outdated or duplicate information input. A research carried out by Data Warehouse Institute and Dun & Bradstreet shows that poor quality information in areas such as marketing, logistics and customer services cost United States economy more than 611 billion dollars in 2002 [5]. Despite the negative indices, there is still some resistance to the adoption of Data Quality processes, being one of its main obstacles the high cost of project implementation. Generally, the deployment of solutions in a company environment is highly demanding, mainly due to the inherent complexity of the Object Identification problem [1]. In addition, it requires high-cost infrastructure, acquisition and integration of additional software.

In this context, our company does research and development of Data Quality and Precision Marketing services, such as customer base quality management systems. Among the products developed by our company, there are a set of Data Quality management services for correcting and preventing incomplete, incorrect, and duplicate data, converting them into consistent and high added value information. In the course of Data Quality services' development, we have been pursuing the reduction of its implementation costs, and in order to achieve this goal, we are now adopting Machine Learning solutions to identify and learn different contexts and recognize specific abnormalities of each client.

Address Validator (AV) is one of the Data Quality modules that takes advantage of the adopted strategy. The address domain is of high complexity, primarily due to the strong interdependency between its elements, and to the variety of possible noise classes. Because of such characteristics, the combinations of problems increase, and databases from different customers may present very specific patterns.

Address Validator (AV) deals with Data Quality problems in the address domain in client databases. AV standardizes, enriches and validates Brazilian addresses, covering different syntactic structures. To support learning, the qualification process carried out by the module was parameterized by a set of numerical values. Even though AV can be adaptable to different address patterns, there is still the need to find the best set of parameters to process this data. Although it seems simple, the parameters are highly interdependent, making it impossible to be manually configured.

In order to minimize the cost of identifying parameters for address qualification, Optimizer was developed, based on the Particle Swarm Optimization (PSO) technique. A specific assessment of the learning process and its efficiency compared to a specialist is presented in [3]. Together, Optimizer and AV enable: 1) fast customizations based on customer specifications, 2) more accurate address qualification, 3) faster deployment, and 4) less human intervention during the process. This happens since the process of defining the parameters is performed automatically while the most common errors present in each client's databases are learned. The difficulty here is that Optimizer needs addresses with patterns of errors to be learned, and a template with the correct format for these data. This implies two major problems: 1) in most cases, the use of real-world databases is constrained by confidentiality issues regarding customer data, and 2) the generation of templates from a real-world dataset is usually a manual, expensive and highly error prone process.

In situations where real-world data are not readily obtained or where specific characteristics need to be represented, automatically generated data can be considered a possible solution. However, to simulate noise, some issues should be taken into consideration, such as: distribution of noise in the database, distribution of noise by fields, constraints between noises (noises that can't occur together in the same field), address classes by assigned noises, etc. Simulating noise and controlling its constraints make the task of building a synthetic dataset very complex. To deal with this issue, a synthetic address data generator was developed. Synthetic Address Data Generator (SADG) automatically generates damaged datasets, as well as their corresponding template, through a flexible representation of rules in XML.

Given the complexity of the address domain, this article aims to assess the ability of the SADG tool to simulate real addresses databases. The assessment is carried out by running the learning process on real and synthetic datasets, and later comparing the results.

BACKGROUND

Organizations all over the world suffer the impacts of poor quality information. The lack of a proper solution to handle inconsistent, incomplete, outdated or duplicate information causes great financial losses. Poor quality information results in costs including rework and income losses, reaching between 10% and 25% [5].

One difficulty in the development of information quality solutions is the lack of available real-world data that characterize the problem in question. Thus, Data Quality tools can benefit from the use of synthetic data generators, for example, data standardization and validation solutions, and applications for detection and elimination of duplicate data. Synthetic data generators are also useful for Information Technology Industry (IT Industry) applications, including *regression testing*, *secure application development*, and *testing of data mining applications* [6].

Systems that generate different types of synthetic data according to their purpose are found in the literature. However, the targeted domain and the characteristics one need to represent so that the synthetic data is considered similar to real-world data may demand more specialized generators. [7] describes a system that generates synthetic rainfall and runoff data, which can be used in the evaluation of rainfall-runoff response sensitivity. [16] presents a system that generates synthetic data which are based on empirical models of experimental data, and uses this data for sensor network evaluation. Due to the lack of real-world data for training sign language recognition systems, [8] developed a synthetic sign language data generator.

Interesting to point out are the systems that generate synthetic data of occupancy similar to real-world data [5, 15]. The Parallel Synthetic Data Generator (PSDG) system described in [6] is designed to generate synthetic data across multiple processors. According to the authors, there is a need for systems that generate "industrial sized" (e.g., terabyte) data sets. Using cluster/grid computing, PSDG quickly generates large amounts of synthetic data. It is worth noting that PSDG focuses mainly on the geographical distribution of occupancy data, such as the distribution of *offices* in the U.S. according to data from *2000 U.S. Census*[1], including *state*, *city*, and *ZIP code* information, among others.

[15] presents the Synthetic Occupancy Generator (SOG) system for synthetic residential occupancy histories (*name* and *address*) generation that can be used in research and processes involving Entity Resolution (ER)[2]. However, SOG does not aim to completely simulate the individuals' behavior, but only those aspects necessary for the generation of realistic occupancy histories.

According to the authors, "residential occupancy", or "occupancy", encompasses someone's personal information (*first name*, *middle name*, *last name*, etc.) and address (*street number*, *street name*, *city*, etc.) over a given period of time (*start date of occupancy*, *end date of occupancy*). In addition, occupancy records may contain additional attributes about someone, such as *date-of-birth*, or about her or his address, such as *telephone number*. "Occupancy history" or "change-of-address" (COA) history is a set of chronologically-consecutive occupancy records for the same individual over a period of time. SOG can improve areas such as business and government by providing input for tools that, for example, identify the same client from different kinds of contact (*address*, *phone*, *email*, etc.).

Generally, synthetic data generators can boost the development in research fields where experiments and assessments need to be carried out with real data. Synthetic data generators are a way to solve problems that arise when privacy issues concerning customer data information constraint the use of real data. The present work uses the Synthetic Address Data Generator, which generates synthetic data similar to

[1] www.census.gov

[2] Entity Resolution (ER) is the process of linking records that reference the same or related real-world entities.

Brazilian addresses real data for Data Quality solutions.

DATA QUALITY SOLUTIONS

This article presents solutions to quality problems in customer data, which require datasets that represent Brazilian real addresses. In the given context, our company does research and development of Data Quality and Precision Marketing services. Among the Data Quality services stands the Address Validator (AV), which solves address data quality problems in customer datasets.

This section presents an overview of the AV service, followed by a description of a solution for the optimization of the AV parameters. Finally, the synthetic address data generator used in experiments with AV is presented.

Address Validator

Address Validator (AV) standardizes, corrects and enriches addresses data. It deals with Data Quality problems in the domain of Brazilian addresses, benefiting customers in several ways, including: improvements in customer relationships, decrease in operational costs, resources optimization and decrease in cash flow problems.

AV can be used in any environment that supports Java without DBMS licensing costs. It supports corporate services in several architectures and allows value adding to legacy systems at a low integration cost. Besides, it can be executed in parallel over multi-core machines as well as over distributed environments [12].

AV can receive as input unstructured, semi-structured and structured data, and returns structured data as output. In the present paper, we work with semi-structured input data, which is understood as input that can have *street name*, *street number*, *secondary identifier*, and *district* in a single field, as illustrated in Figure 1. In this example, the corresponding output presents the following structured information: *suffix*, *street name*, *street number*, *secondary identifier*, *district*, *ZIP code*, *city* and *state*. Table 1 describes the qualified information and the corresponding address classification returned by AV (see *Classification*).

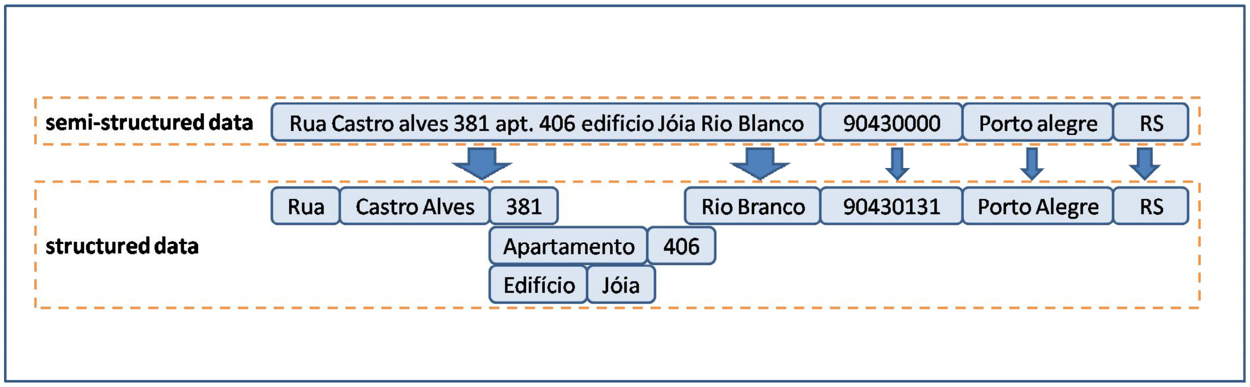

Figure 1: Data Structure

There are four classes returned by AV, related to the classification of addresses: i) *Valid*: input address is correct, being only validated by the service; ii) *Qualified*: input address has some incorrect field, which was inferred or corrected by the service; iii) *Not Qualified*: input address can't be qualified because of missing fields or inconsistencies; iv) *CEP5:* input address has a generic *ZIP code*[3]. Back to Table 1, AV returns the class *Qualified* for the given address example, because the *ZIP code* field was corrected (from *90430000* to *90430131*).

Address Data	**Example**
Suffix	Rua
Street name	Castro Alves
Street number	381
Secondary identification	Apartamento 406 Edifício Jóia
District	Rio Branco
ZIP code	90430131
City	Porto Alegre
State	RS
Classification	Qualified

Table 1: Example of AV Output

Brazil is a federal republic comprising 26 federal states[4] and a Federal District[5] (*DF*), where the Brazilian capital, Brasilia, is located. It should be noted that Brazilian addresses do not follow a single addressing structure. Most localities follow the same pattern. However, some places have their own specific pattern, called here complex structures. *DF* addressing structure can be considered one of the most important special cases, basically because of two points: 1) its great political and commercial importance and 2) its structure also occurs in other cities. For better understanding of these more complex addresses, we will use Brasilia addressing structure in our examples.

Brasilia is a planned city architected by Oscar Niemeyer. When seen from above, the layout of the city resembles the shape of an airplane or a dragonfly. Brasilia is divided in areas for residence, business, schools, churches, etc., and the streets and roads are identified not by names, but by acronyms and numbers arranged according to a geographical system of *Blocks* (Quadra) and *Sectors* (e.g.: *SBS – South Bank Sector*, *SIG - Graphic Industry Sector,* etc.). This structure differs from most Brazilian federal states addresses, and its complexity makes the qualification of the *DF* addresses difficult. In order to illustrate such differences, Table 2 shows two examples of Brasilia addresses.

Id	**Street name**	**District**	**ZIP code**	**City**	**State**
1	SBS QUADRA 4 *(SBS BLOCK 4)*	ASA SUL	70070140	BRASILIA	DF
2	SIG QUADRA 6 LOTE 800 (*SIG BLOCK 6 LOT 800*)	ZONA INDUSTRIAL	70610440	BRASILIA	DF

Table 2: Examples of Brasilia addresses

Another characteristic of *DF* addresses is related to changes in abbreviations to refer to specific addresses. These changes, however, are not the most obvious, and they are usually shared/known only by the city citizens, thus making the identification of the targeted address more difficult. As an example, the following address, taken from a real database, presents the characteristic "*SHC/CL SUL QUADRA 306*" (*SHC/SOUTH CL BLOCK 306*) meaning "*CLS BLOCO 306*" (*CLS BLOCK 306*), where "SHC/SOUTH CL" corresponds to "CLS".

For address qualification, AV uses a knowledge base that contains Brazilian addressing elements, called

[3] For each small city, Brazil has a single ZIP code shared by all addresses in that city.
[4] http://en.wikipedia.org/wiki/States_of_Brazil
[5] http://en.wikipedia.org/wiki/Brazilian_Federal_District

Dynamic Address Database (DAD), which covers Brazilian addresses, including the ones from *DF*. This base is used as reference for the AV addresses qualification. Back to the example, by querying DAD, we find that the correct/expected address is *"CLS BLOCO 306" (CLS BLOCK 306)*.

Indeed, the AV service can deal with more complex addressing structures, such as *DF* addresses, as well as simpler structures adopted by other Brazilian cities. AV makes the inclusion of any address structure easier, because it does not require syntactic rules to the formation of addresses. Due to this feature, AV is fully adaptable to different address data patterns, although there is the need to find the best set of parameters to process this data. The main parameters among the ones used by AV are the following:

- *Match*: Parameter that indicates match relevance by field for the qualification. The fields used are *suffix*, *title name, street name*, *district*, *ZIP code*, *city* and *state*.
- *Order*: Parameter that indicates the order of occurrence of the fields that is preferred for a given dataset. E.g. the *street name* field followed by the *district* field.
- *Tokens Usage:* Parameter that indicates the number of tokens from an input record matching the addresses reference database (DAD). Back to the example of Table 1, the input *"Rua Castro Alves 381 Apartamento 406 Edifício Jóia"* results in *Tokens Usage* = 0,5 (of 8 input tokens, only 4 match DAD, because the content of *secondary identification* does not match).

It should be pointed out that there is a high dependence between the 33 numeric parameters that should be set prior to AV execution, which makes it impossible for them to be set manually [3]. To solve this problem, an application that optimizes the AV parameters was developed in order to find the best parameter set used in the system's qualifying process.

Optimizer

For the AV service to handle different address structures, we need to set its parameters. As previously stated, there is a high dependency among the AV parameters, making it harder to set them manually. This set of parameters may be represented as a vector, making the application of some sort of optimization method easier.

Optimization problems are frequent in many branches of knowledge facing difficulties to find out better or more suitable processes or methods. Generally, optimization problems consist of maximizing or minimizing a function defined over a given domain. Optimization theory involves the task of determining which of the existing solutions for a certain problem is the best one. In other words, which of them is the optimal solution. Algorithms based on Evolutionary Computing techniques are proving to be very efficient in the search for optimal solutions in a wide variety of problems [13].

The application of the Particle Swarm Optimization (PSO) algorithm in the Address Validator (AV) service in order to find the best set of parameters necessary to correctly qualify addresses is presented next.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary computational technique for global optimization, which is motivated by the simulation of social behavior that underlies the movements of the swarm of insects or birds' flock [9]. It was developed by Kennedy and Eberhart, and consists of the optimization of a fitness function by information exchange between the individuals (insects or birds) of a given population (swarm or flock). It is worth mentioning that during the search for food or nest the individuals, called particles, use their experiences, as well as the experiences of the whole population, to find the optimal solution. This way, particles learn from each other's success.

The PSO algorithm is basically composed by a velocity vector and a position vector. The position of each particle is updated according to the current velocity, the experience acquired by the particle, and the experience acquired by the population. Each particle stores its current position and velocity, its current fitness value, its best position, and its best fitness value so far. In addition, each particle is aware of its neighbors, and knows as well the best location and best fitness function value among them. Thus, the PSO algorithm should know the best global position, which denotes the position of the best particle in the population, i.e. the particle which is closer to the optimal solution.

PSO is initialized with a group of random particles, describing potential solutions. It then searches the optimal result by updating the population of particles. Every particle has a fitness value resulting from its evaluation by a fitness function, which should be optimized (maximized or minimized), and a velocity that directs its path. Particles fly in a given problem space following those particles that achieve the best results. At each iteration, each particle is updated according to two values: the value of the best solution (fitness) found by the particle so far and the value of the best solution found by any particle of the population so far, which is called global best.

In some scenarios, like the one presented here, the particles do not know where the best solution is, but they do know how far it is at each iteration. The strategy is then to follow the particle that is closer to the solution. After finding those two values, the particle updates its velocity and position according to Equations 1 and 2, respectively [9]. The best position is the one in which the particle achieves the best result for the fitness function.

$$v[] = v[] + c_1 * rand() * (localBest[] - p[]) + c_2 * rand() * (globalBest[] - p[]) \qquad (1)$$

$$p[] = p[] + v[] \qquad (2)$$

In Equation 1 and 2, *v[]* represents the particle velocity, c_1 and c_2 are the learning factors, *rand()* denotes a random number between 0 and 1, *localBest[]* represents the best particle position so far, *globalBest[]* represents the best position in the whole population of particles so far, and *p[]* is the particle current position.

Improvements in the PSO algorithm components are usually found in the literature [2, 4, 14], and they aim to improve the PSO convergence velocity by changing the velocity update equation without changing the structure of the algorithm itself. One of them is the addition of an Inertia Weight W in Equation 1 [14], which controls the algorithm exploration ability and therefore improves the speed with which the particles find the optimal solution (see Equation 3).

$$v[] = W * v[] + c_1 * rand() * (localBest[] - p[]) + c_2 * rand() * (globalBest[] - p[]) \qquad (3)$$

$$v[] = k * (v[] + c_1 * rand() * (localBest[] - p[]) + c_2 * rand() * (globalBest[] - p[])) \qquad (4)$$

$$k = {}^{2}\!\big/_{\left|2 - \varphi - \sqrt{(\varphi^2 - 4\varphi)}\right|} \text{ where } \varphi = c_1 + c_2,\ \varphi > 4 \qquad (5)$$

Another improvement that helps the convergence of the PSO algorithm is the use of a Constriction Factor, which proposes a new method for choosing the Inertia Weight W and the learning factors c_1 and c_2, so that the convergence is guaranteed [2]. Equation 4 shows the assimilation of this factor, where k is a function of c_1 and c_2, as reflected in Equation 5 [4].

Optimizer Overview

Optimizer is a module developed for training the AV service and other Data Quality services. In this paper, we focus in address services. Optimizer's goal is to find the best parameter set used in the system's qualification process. Figure 2 illustrates the Optimizer process.

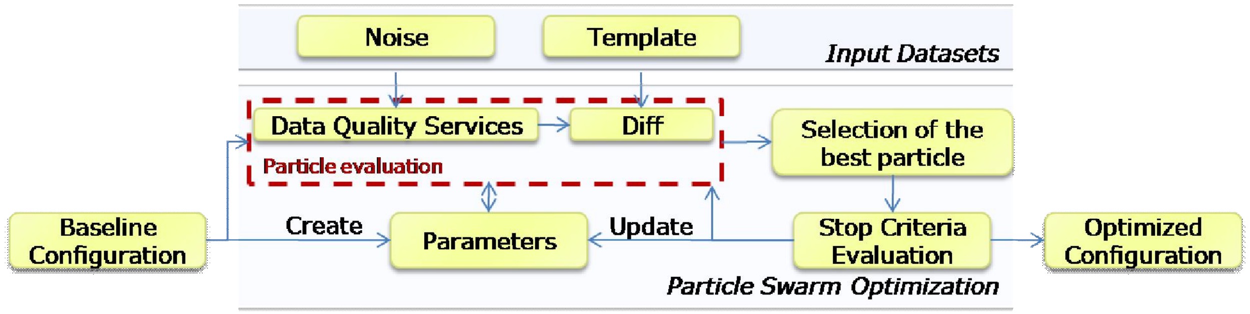

Figure 2: Optimizer process

For the AV training process using the Optimizer module, two input files are necessary: a file with noisy address data (Noise) and a file with correct address data (Template). Tables 3 and 4 illustrate examples of noisy addresses included in the Noise file, and the corresponding correct addresses in the Template file, respectively.

Id	Address	District	ZIP code	City	State
1	RUA NOBERTO WON GAL 48	REDENCAO	69049100	MANAUS	AMM
2	AV RORARY 000667	FAROL	57050480	MACEIO	ALL
3	SHCN COM LOCAL QUADRA 403 BLOCO D	ASA NORTE	70310500	BRASILIA	DF

Table 3: Example of Noise file

Id	Suffix	Street name	Street number	District	ZIP code	City	State
1	RUA	NOBERTO VON GAL	48	REDENCAO	69049100	MANAUS	AM
2	AVENIDA	ROTARY	667	FAROL	57052480	MACEIO	AL
3		CLN QUADRA 403 BLOCO D		ASA NORTE	70835540	BRASILIA	DF

Table 4: Example of Template file

The examples of the Noise file shown in Table 3 were taken from real-world datasets and contain different types of noises. The corresponding Template file, however, required correction (Table 4), which is usually manually performed, being a laborious, time consuming and error prone process.

Building these files is a bottle neck in the process, and this is the reason behind the development of a synthetic data generator that produces both files by applying rules over a knowledge base including Brazilian addresses (DAD).

Thus, the former file works as input for the execution of AV, and the later file works as a template for the evaluation of this input data qualification (see Figure 2, where the mentioned files are used by the *Input Datasets* component). Both files are used in the AV training process.

As previously described, every particle of the population must be evaluated at each iteration of the PSO processing. Considering the AV service, the evaluation of each particle consists of executing AV using the particle current position values as its parameters. AV processes the input noisy addresses, qualifying and classifying them into certain classes (*Data Quality Services* - Figure 2).

The output generated by AV is compared by the Diff component of Figure 2 against the template addresses using their classification and qualification. For such, the Diff component generates a confusion matrix which matches correct against predicted classifications, i.e. the expected output class for a given address described in the template compared to the class outputted by AV. It is also possible to determine which addresses were correctly qualified, where the correct qualification is the inference of an exact candidate for a given noisy address. In order to do that, the address returned by AV is compared field by field against the corresponding address in the template.

Based on this comparison, the execution of AV is evaluated by a fitness function (*Selection of the best particle* component - Figure 2). The four possible fitness functions provided by Optimizer are: *simple fitness function*; *fitness function using thresholds and weights for each record field*; *fitness function taking correct records into account*; *fitness function taking into account correct records classified into wrong classes*, which are described in [3].

The complete Optimizer process ends when a stop criterion, set beforehand, is achieved. Besides the standard configurations of the PSO approach, Optimizer also allows customization for fine-tuning the algorithm. The specific configurations of Optimizer include: *number of threads*; *stop criteria*; and *particles update type* [3]. Here we use the Optimizer configurations presented in [3] in the AV training process.

Synthetic Address Data Generator

Synthetic Address Data Generator (SADG) is a tool for generating synthetic Brazilian addresses' data with the characteristics of real-world databases from this domain. Generally, these characteristics represent the most common errors found in real addresses' datasets.

SADG is implemented in Java. In order to describe SADG flow, we developed a XML[6]– based language that enables to flexibly describe the synthetic addresses generation process. This language required the definition of different constraints in the rules for noise application, because the address domain is quite complex, which is primarily due to two reasons: 1) high interdependence between the address elements, 2) wide variety of possible noise classes. An example of the complexity in building up synthetic addresses is the following: an address where *street name* is incomplete and *ZIP code* is incorrect. In this case, the address has two different types of noises that directly impact on qualification. The correct qualification for this address depends on deciding which element will be considered more reliable: *street name* or *ZIP code*. This decision should be guided by the error pattern of the real-world dataset to be represented.

Considering this context, we used *Interpreter pattern* and *Specification pattern* to generate a Boolean expression that defines the application of the rule, thus making the solution extensible, easy to change and add new constraints. Note that SADG language is not limited to the addresses domain, and can be easily configured to another application domain.

[6]eXtensible Markup Language

SADG encodes the classes of address' noises based on configurable rules. Moreover, it generates different address structures, and supports the inclusion of structural noises. As output from SADG we have a file with the noisy addresses (Noise) and a file containing the corresponding correct addresses (Template). These files consist of address structures similar to those previously presented in Tables 3 and 4, respectively. Figure 3 illustrates the process of the SADG tool.

Starting the SADG process requires a reference dataset that provides information to the SADG stages (*Data Source* – Figure 3). As previously mentioned, DAD is the reference database of Brazilian address.

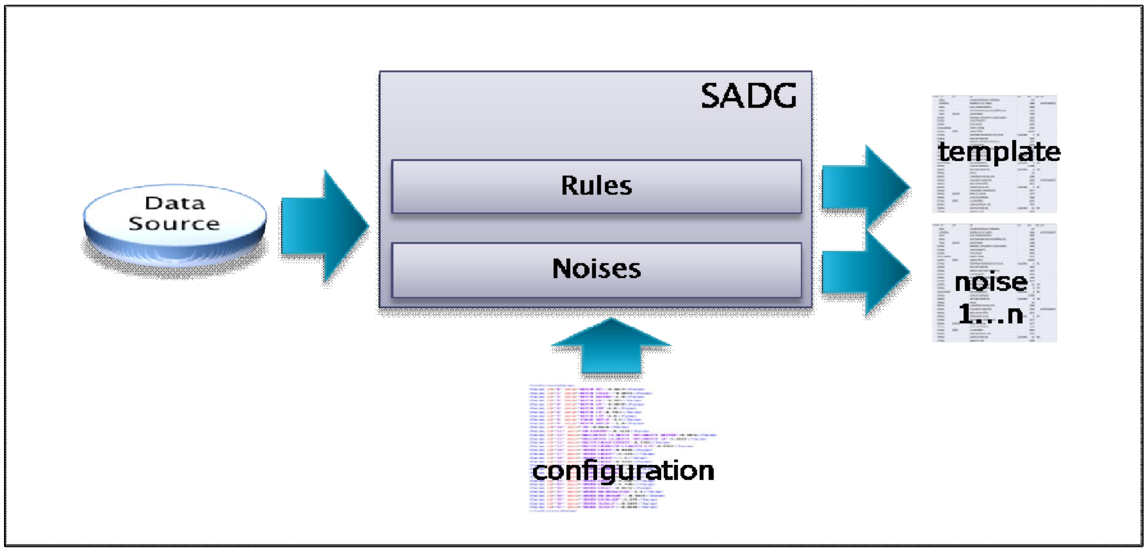

Figure 3: SADG process

With the SADG language, the following information is coded in the *Configuration* file (see Figure 3):

- Percentages of addresses selected from DAD to generate the output files as well as the distribution of these addresses in the 26 Brazilian federal states and in *DF*;
- How and which information from DAD will be used in the synthetic data generation process;
- Different noise classes (*Noises*) applied to the addresses selected from DAD;
- The rules for the application of noise classes (*Rules*) together with constraints to the application of these rules;
- The order of the rules' execution and the dependencies between the application of different rules on the same address;
- Reference to external dictionaries to support the rules for the application of noise classes;
- How the output files will be structured, considering that, in this work, the Noise file will contain semi-structured addresses and the Template file will contain structured addresses.

A study on the possible errors found in real-world datasets of Brazilian addresses was carried out, which enabled defining the noise classes currently available in SADG. Among these classes, we can point out:

- *Typographical errors:* including address misspelling, such as characters insertion, removal and transposition; and application of similar phonetics in addresses.
- *Missing terms:* eliminating the constituent elements of an address, e.g., the *suffix* of a given address.
- *Term permutation:* applying permutations of the constituent elements of an address, e.g., the *suffix* preceded by *street name*.

- *Consistent but incorrect data:* changing the value of an address element by another value that is valid but incorrect for this address. For example, for a given address with the *suffix Street*, change the value of this element to *Avenue*.
- *Abbreviation:* inclusion of abbreviations for address elements, e.g., *B* to *Block*.
- *Out of context words:* inserting words out of the address context, such as bad words in a given address.
- *Term duplication:* applying term duplication, such as repetition of *suffix* in a given address.

For generating the presented noise classes, a set of rules was developed, which took into consideration the percentages of each noise application as well as the constraints needed for their correct application. The SADG language enables adding new noise classes by extending Java classes, and gives the flexibility to create and to configure the corresponding application rules that implement such noises.

Table 5 shows an example of the *typographical errors* rule being applied to an address taken from DAD. Note that *typographical errors* was applied to the given address *suffix* and *street name* (*address* column), *district* and *city* information, respecting the *ZIP code* constraint, defined in the *typographical errors* rule[7] (correct input *ZIP code* – 95096170).

Id	**Address**	**District**	**ZIP code**	**City**	**State**
1	URA ALFREDU MILLANI 28	EZPLANADA	95096170	CAIXAS DO SUL	RS

Table 5: Example of *typographical error* – Noise file

Id	**Suffix**	**Street name**	**Street number**	**District**	**ZIP code**	**City**	**State**
1	RUA	ALFREDO MILANI	28	ESPLANADA	95096170	CAXIAS DO SUL	RS

Table 6: Example of Template file

At the end of the SADG process, the noisy address (Table 5) will be stored in the Noise file, and the corresponding correct address (Table 6) will be stored in the Template file. SADG resulting files will be used in the AV service experiments.

RESULTS

This section presents the results of the Optimizer execution over the AV. The experiments were carried out according to the following steps: 1) AV training by Optimizer, 2) AV testing with new data, using the resulting configuration from step 1.

Datasets

The datasets used for the experiments were chosen based on the two main syntactic structures that Brazilian addresses represent, both affecting precision in qualification. Those datasets were classified as complex structure addresses and simple structure addresses.

The address databases with complex structures we use contain records only from *DF,* since they share a specific syntactic structure and consequently need special treatment for their qualification. Address

[7] In this case, the constraint says that the *typographical error* rule will be applied only to input addresses without noise in *ZIP code*, that is, records where the input *ZIP code* is correct.

datasets with simple structures contain records covering the 26 Brazilian federal states, here called *Standard* addresses.

In summary, we used datasets with both real and syntactic addresses for the experiments. Records distribution for the two datasets with real addresses is the following: 306 records from *DF* and 400 from *Standard*. Datasets with syntactic addresses also have the same distribution (306 records from *DF* and 400 from *Standard*), but differ from the records in the real-world datasets for being generated by the SADG. These four datasets used in our experiments with the AV service are described next.

Experiments and Discussion

In this work, training experiments of the AV service were conducted using the Optimizer in order to find the best parameters for the qualification of Brazilian addresses. Knowing the best AV parameters' values, found out by the application of the Optimizer, tests with this parameters set were carried out to assess the qualification of the new addresses.

The training experiments were designed to verify whether two sets of parameters for AV, generated from the training with a real database and from a synthetic dataset, are able to achieve similar results when tested in the process of qualification of a synthetic dataset and of a real database, respectively. In such a context, we chose to run the experiments considering the following two cases:

- *Case 1:* training with real address databases (*DF and Standard*) and testing with synthetic address sets (*DF* and *Standard*);
- *Case 2:* training with synthetic address databases (*DF* and *Standard*) and testing with real address sets (*DF and Standard*).

From the training experiments, we aim at verifying whether SADG is able to generate synthetic data similar to the real address data. Therefore, the results from *Case 1* should be close to those achieved in *Case 2*. Results from the training experiments with AV are presented in Table 7 and graphically illustrated in Figure 4. The results were measured by Precision, Recall, and the harmonic mean (represented by F-measure [10]).

Datasets	**Measure**	**Case 1**	**Case 2**
Standard	Precision	75.47%	81.08%
	Recall	93.68%	93.33%
	F-measure	83.59%	86.77%
DF	Precision	62.18%	69.75%
	Recall	94.87%	96.69%
	F-measure	75.12%	81.03%

Table 7: AV results

The results from AV for *Standard* addresses reached an F-measure of 83.59% for *Case 1* and 86.77% for *Case 2*, which are very close values. We point out a difference of less than 6% in the Precision (75.47% and 81.08% for *Case 1* and *Case 2*, respectively). Similarly, the results from AV for the *DF* addresses reached close values for the performance measures shown in Table 7: F-measure of 75.12% for *Case 1* and 81.03% for *Case 2* (a difference of around 5%) and Precision of 62.18% for *Case 1* and 69.75% for *Case 2* (a difference of around 7%).

For the performance assessment of *Case 1* against *Case 2* we used the significance test [11], calculating the relevance between the results by means of statistics rather than a subjective criterion. In the present work, the objective of applying this test is verifying whether *Case 1* and *Case 2* are behaving similarly in both addresses scenarios: simple structures (*Standard*) and complex structures (*DF*). In general, the test results displayed non-significant levels (less than 95%), indicating that the synthetic datasets were able to simulate the characteristics of real addresses. Thus, we confirm the similarity of synthetic and real databases in both cases.

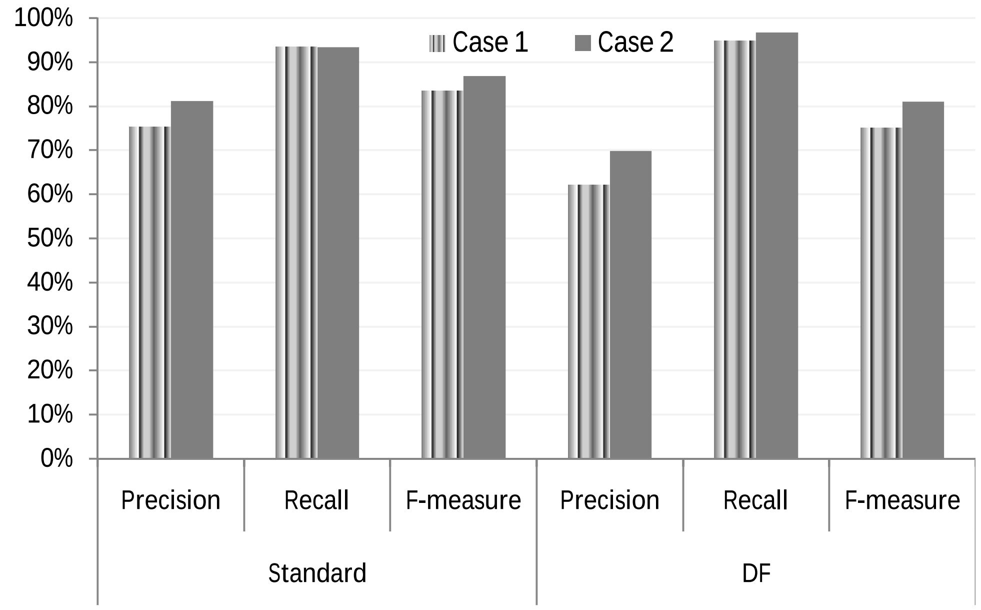

Figure 4: Graphic chart for the results of AV tests

The present assessment shows that synthetic address datasets generated by SADG can be successfully used by applications that need addresses that represent specific characteristics of real data.

CONCLUSIONS

This paper presented the AV service, an application for Data Quality problems encompassing different structures and noise types contained in the Brazilian addresses domain. The Optimizer module, which is based on the PSO approach, was used in the optimization of the AV parameters in order to find its best configuration. To that end, datasets of addresses representing different quality problems were necessary. Given that need, adding to the absence of available real-world databases and the difficulty in creating the corresponding templates, synthetic datasets simulating real addresses with simple and complex structures were generated by the SADG. Test results with AV proving that synthetic datasets were able to simulate real addresses were shown.

Integrating these modules, our solution goes full circle (SADG → Optimizer → AV), enabling fast deployment of its services regardless of the characteristics to be dealt with for each client.

Due to the difficult production of a compact dataset that covers all kinds of address noises for the training of AV, an incremental learning is intended to be carried out as a future work, where it will be possible to add new address patterns to the training data in the optimization process.

Although we have the benefits described in this paper, our approach does not deal with dynamic contexts, as business rules changes. To solve this problem we will research and develop a module that can detect specific abnormalities in client datasets. The initial idea is to use some of the qualification information provided by AV to represent the distribution and noise classes and then, by using SADG, generate synthetic datasets with such characteristics, later to be used for training the Optimizer.

ACKNOWLEDGEMENTS

The present work is supported by FINEP[8] (project number: 01.07.0178.00) and Acxiom Brazil[9].

REFERENCES

[1] Batini, C. and Scannapieco. M. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[2] Clerc, M. "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization". *Proceedings of the IEEE Congress on Evolutionary Computation*. 1999. pp. 1951–1957. Washington, DC.

[3] De Abreu, S. C., Junior, L. C. R., De Mello, M. T. and De Almeida, T. O. "Data Quality enhancements supported by a Particle Swarm Optimization Approach". *Communications of SIWN - The Systemics and Informatics World Network*, 2010.

[4] Eberhart, R. C. and Shi, Y. "Comparing inertia weights and constriction factors in particle swarm optimization". *Proceedings of the 2000 Congress On Evolutionary Computation*. 2000. pp. 84–88. San Diego, CA.

[5] Eckerson, W. W. "Data quality and the bottom line: Achieving business success through a commitment to high quality data". *The Data Warehousing Institute*. 2002. pp. 1–36.

[6] Hoag, J. E. and Thompson, C. W. "A parallel general-purpose synthetic data generator". *SIGMOD Record*, 36 (1). 2007. pp. 19-24.

[7] Hromadka, T. V. "A rainfall-runoff probabilistic simulation program". *Environmental Software*, 11 (4). 1996. pp. 243-249.

[8] Jiang, F., Gao, W., Yao, H., Zhao D. and Chen, X. "Synthetic data generation technique in Signer-independent sign language recognition". *Pattern Recognition Letters*, 30 (5). 2009. pp. 513-524.

[9] Kennedy, J. and Eberhart, R. C. "Particle swarm optimization". *IEEE International Conference on Neural Networks*. 1995. pp. 1942–1948, Perth, Australia.

[10] Kowalski, G. *Information retrieval systems: theory and implementation*. Kluwer Academic Publishers, Boston, 1997.

[11] Manning, C.D. and Schutze, H. *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, 1999.

[12] Martini, A., Frainer, G. C., Rios, R. A., Pereira Jr., L. A., Storch, M., Geiss, M. and Mello, R. F. "Gostorm: A new platform driven to scale-out environments". *The 3rd International Workshop on Latin American Grid, IEEE Computer, (1)*. 2009. pp. 1-6. São Paulo, SP, Brasil.

[13] Rezende, S. O. *Sistemas Inteligentes: Fundamentos e Aplicações*. Editora Manole, Barueri, SP, Brasil, 2005.

[14] Shi, Y. and Eberhart, R. C. "Parameter selection in particle swarm optimization". *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*. Springer-Verlag. 1998. pp. 591–600. London, UK.

[15] Talburt, J., Zhou, Y. and Shivaiah, S. Y. "SOG: A Synthetic Occupancy Generator to Support Entity Resolution Instruction and Research". *Proceedings of the 14th International Conference on Information Quality (ICIQ 2009)*. 2009. Hasso Plattner Institute, University of Potsdam, Germany.

[16] Yu, Y., Ganesan, D., Girod, L., Estrin, D. and Govindan, R. "Synthetic Data Generation to Support Irregular Sampling in Sensor Networks". *Geo Sensor Networks*. 2003.

[8] http://www.finep.gov.br/
[9] http://www.acxiom.com.br/