Human Drivers Knowledge Integration in a Logistics Decision Support Tool

María D. R-Moreno, David Camacho, David F. Barrero, and Bonifacio Castaño

Abstract. The logistics sector is a very dynamic and complex domain with large impact in our day-to-day life. It is also very sensitive to the market changes, so tools that can improve their processes are needed. This paper describes a decision support tool for a Spanish Logistics Company. The tool takes the knowlege from the human drivers, i.e. *the address used to deliver a package*, integrates and exploits it into the decision process. However, this is precisely the data that is often wrongly introduced by the drivers. The tool analyses and detects different mistakes (i.e. misspellings) in the addresses introduced, groups by zones the packages that have to be delivered, and proposes the routes that the drivers should follow. To achieve this, the tool combines three different techniques from Artificial Intelligence. First, Data Mining techniques are used to detect and correct addresses inconsistencies. Second, Case-Based Reasoning techniques that are employed to separate and learn the most frequent areas or zones that the experienced drivers do. And finally, we use Evolutionary Computation (EC) to plan optimal routes (using the human drivers past experiences) from the learned areas, and evaluate those plans against the original route executed by the human driver. Results show that the tool can automatically correct most of the misspelling in the data.

María D. R-Moreno · David F. Barrero
Computer Engineering Department. Universidad de Alcalá, Madrid, Spain
e-mail: `(mdolores,david)@aut.uah.es`

David Camacho
Computer Science Department. Universidad Autónoma de Madrid, Madrid, Spain
e-mail: `david.camacho@uam.es`

Bonifacio Castano
Mathematics Department. Universidad de Alcalá, Madrid, Spain
e-mail: `bonifacio.castano@uah.es`

F.M.T. Brazier et al. (Eds.): Intelligent Distributed Computing V, SCI 382, pp. 227–236.
springerlink.com © Springer-Verlag Berlin Heidelberg 2011

1 Introduction

The need for precise and trustable information in the logistics sector has driven the development of complex Geographic Information Systems (GIS), very useful for planning their routes. Those systems are combined with Global Positioning Systems (GPS) for positioning the different elements involved in the shipments. However, those systems although very useful, cannot make decisions based on, for example, shortcuts in the route that human operators (drivers) learn from the experience.

In our particular problem, the logistics company has a set of logistics distributors with General Packet Radio Service (GPRS) connexions and Personal Digital Assistants (PDAs) where the list of shipments is stored for each day. The list of tasks for each distributor is supplied each day from a central server. The order in which they should be carried out is decided by the driver depending on the situation: density of the traffic, state of the highway, hour of the day and place of the delivery. Although the final decision could depend by other topics such as the weather conditions or simply some kind of intuition, it will be finally taked by the driver based on his previous experiencies. However, this essential knowledge cannot be easily represented, handled and incorportated in a logistics tool.

Therefore, from the daily interaction between human drivers and the logistic tool, two main issues can be learned: the particular route that the drivers followed and the addresses that was delivered by the driver. In our previous work [10], it was shown how a particular representation for the drivers route can be used to be stored in a database of plans, and how they can be retrieved and later reused to plan new routes. However, from these initial results we detected that the main problem of reusing the drivers knowledge was related to the mistakes that appear in the addresses introduced. It avoids the system to correctly reuse the available knowledge.

Within this context, the company needs a decision support tool that, on the one hand, allows them to correct mistakes in the delivery addresses anotated by the users.And on the other hand, the company needs a tool to help them to identify which are the best drivers behaviors, learn from them and plan optimal routes.

In this paper we present a tool that combines techniques from three different AI areas to solve the above problems. First, Data Mining techniques to detect and correct inconsistencies in the addresses introduced by the drivers. Second, Case-Based Reasoning (CBR) techniques to separate and learn the most frequent areas or zones that the experienced logistic operators do. Third, we use Evolutionary Computation (EC) to plan optimal routes from the learning areas and evaluate them [10].

The rest of the paper is structured as follows. Section 2 presents an overview of the decision support tool. Section 3 describes the Record Linkage module developed to eliminate misspellings in the input addresses dataset. Section 4 presents the CBR and then section 5 describes its intregation into an Evolutionary Computation (EC) module used for route optimization. Section 6 shows an experimental evaluation of our application with the real data provided by the logistic company. Finally, some conclusions are outlined.

2 Intelligent Distributed Architecture

There are several available tools that address (partially) our described problem. For example, the ILOG SOLVER provides scheduling solutions in some domains such as manufacturing or transportation and the ILOG DISPATCHER provides extensions for vehicle routing [9]. There is a huge number of commercial and research software applications that provides different (usually *ad-hoc*) solutions to this problem, those are mainly based on the efficiency of the algorithms used [11], [12]. However, none of these tools, algorithms or solutions allow to automatically combine data processing to detect wrong adresses, the experience of the drivers learning from their past decisions, and planning optimal routes from the learnt knowledge as we propose.

Our application has been divided in five subsytems described as follows:

- The *Statistic Subsystem*: takes the data from the files provided by the logistic company in CSV format (Comma Separated Values). It analyses that the files are valid, loads the information into data structures and performs a first statistic analysis of each driver contained in the file. The analysis takes into account the number of working days, number of physical acts, average of physical acts performed, or average of the type of confirmation received by the client.
- The *Automated Data Processing Subsystem*: is based on Data Mining techniques. It takes as input a pair of addresses and it detects matching or differences. Statistics are calculated from the agreement on matching or differing addresses. A threshold is defined by the user, so values above that threshold are automatically replaced. Values below the threshold are shown to the user together to the possible solutions that the algorithm has found.
- The *Loading Data Subsystem*: is in charge of obtaining and loading the geographic information. It takes the latitude and longitude of each address and calculates the distances among right and corrected addresses. This information is provided by means of the Google Maps API. During the process of calculating the coordinates, we group the same addresses because a driver can delivery several packages in the same address. We name that *act*. So, in the database, the addresses will be loaded as *acts*, and an extra field will be added to represent the number of deliveries and pick ups for that act.
- The *Learning Working Areas (LWA) Subsystem*: creates zones or working areas for each driver using the data loaded and processed in the previous subsystems. To generate that subdivision we have used the zip code as the baseline. So, a zone or a working area might be representated using more than one zip code. It also allows us to visualize (using the Google Maps API) the different working areas.
- The *Learning Manager Task (LMT) Subsystem*: plans routes in a date selected by the user. We can specify the number of different plans we want as output, and compare them with the original plan performed by the driver. In the comparison and the generation of plans, we can use different parameters to measure the quality of the plans, such as the total distance, positive rate of LWA, time, etc. The planning algorithm can adapt its answer to the driver behavior and generate plans according to it, if that is what we want.

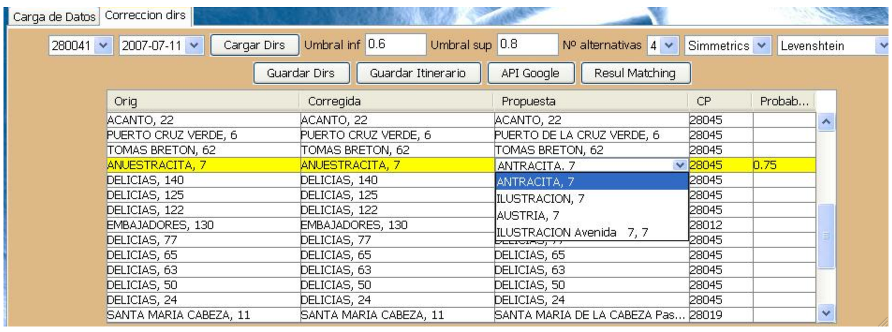
3 Automated Data Processing

Automatic detection of similar or duplicate entities is a critical problem in a wide range of applications, from Web learning to protein sequence identification [3, 8]. The identification of similar entities, or Record Linkage (RL), is necessary to merge databases with complementary information. Then, there is a need to algorithmically solve these problems given the costs of using human expertise.

The problem of RL was defined as the process of matching equivalent records that differ syntactically [8]. In [3], the entity matching issue is formulated as a classification problem, where the basic goal is to classify entity pairs as matching or non-matching, based on statistical unsupervised methods. Edit distances, such as Levenshtein [6] or Jaro-Winkler [13], have been primarily used to solve this problem in the past. A string edit distance is a metric that can be used to determine how closer are two strings. This value is obtained in terms of the number of character insertions, and deletions, needed to convert one into the other.

In our application, the percentage of mispelling in the addresses is around 45%. It is clearly the bottleneck of our application. After some preprocessing the percentage drops to 20%. But this is still very high when attempting to perform an automatic comparison of the routes.So we have implemented different RL methods and compare the results in our domain. We briefly describe them:

- The LEVENSHTEIN distance between two strings [6] is defined as the minimum number of edits needed to transform one string into another one. The allowable edit operations are insertion, deletion, or substitution of a single character.
- The JARO-WINKLER [5] distance is a measure of similarity between two strings. It is a variant of the Jaro distance metric and mainly used in the area of RL. The higher the Jaro-Winkler distance for two strings is, the more similar the strings are. The score is normalized such that 0 equates to no similarity and 1 is an exact match.
- The BLOCK DISTANCE, also known as Manhattan distance, represents distances between points in a city road grid. In the case of strings, it examines the absolute differences between caracters of a pair of strings.
- The COSINE distance is a measure of similarity between two vectors using the Cosine's angle between them. The result of the Cosine function is equal to 1 when the angle is 0, and it is less than 1 when the angle is of any other value. The value of the angles allows us to determine whether two vectors/words are pointing in roughly the same direction (same word).
- The DICE distance can be used for strings grouping two written letters (bigrams) of the 2 strings we want to compare. Once we have the set of bigrams for each string, then the DICE distance is the result of dividing the double of the number of character bigrams found in both strings by the number of bigrams in the first string plus the number of bigrams in the second string.
- The MONGE ELKAN distance [7] uses variable costs depending on the substring gaps between the words. This distance can be considered as an hybrid distance because it can accommodate multiple static string distances using several parameters.



280041 | 2007-07-11 | Umbral inf 0.6 | Umbral sup 0.8 | Nº alternativas 4 | Simmetrics | Levenshtein

Orig	Corregida	Propuesta	CP	Probab...
ACANTO, 22	ACANTO, 22	ACANTO, 22	28045	
PUERTO CRUZ VERDE, 6	PUERTO CRUZ VERDE, 6	PUERTO DE LA CRUZ VERDE, 6	28045	
TOMAS BRETON, 62	TOMAS BRETON, 62	TOMAS BRETON, 62	28045	
ANUESTRACITA, 7	ANUESTRACITA, 7	ANTRACITA. 7	28045	0.75
DELICIAS, 140	DELICIAS, 140	ANTRACITA, 7	28045	
DELICIAS, 125	DELICIAS, 125	ILUSTRACION, 7	28045	
DELICIAS, 122	DELICIAS, 122	AUSTRIA, 7	28045	
EMBAJADORES, 130	EMBAJADORES, 130	ILUSTRACION Avenida 7, 7	28012	
DELICIAS, 77	DELICIAS, 77		28045	
DELICIAS, 65	DELICIAS, 65	DELICIAS, 65	28045	
DELICIAS, 63	DELICIAS, 63	DELICIAS, 63	28045	
DELICIAS, 50	DELICIAS, 50	DELICIAS, 50	28045	
DELICIAS, 24	DELICIAS, 24	DELICIAS, 24	28045	
SANTA MARIA CABEZA, 11	SANTA MARIA CABEZA, 11	SANTA MARIA DE LA CABEZA Pas...	28019	

Fig. 1 Automated data processing interface.

Fig. 1 shows the API where the user can set some aspects of the algorithm such as the upper and lower threshold values he wants to apply. When the value returned by the algorithm is greater than the upper threshold value, the wrong street is automatically replaced by the string with the highest value found by the algorithm. The row that contains the change is marked as green. For values between the upper and lower threshold values, the application returns different alternatives and the user has to decide which one is the correct one. The row that contains this case is marked as yellow . If the value returned by the algorithm is smaller than the lower threshold value, the row is marked as red and no solutions are given. The user can change the lower threshold value or manually correct the street.

In Section 6 a set of experimental results will be shown to demonstrate how these techniques allow one to alleviate the problem of mismatching. The experiments have been carried out over a set of real data to stablish what is the best metric that should be used to our current problem.

4 Planning Routes from Past Experiences

In order to learn good behaviours from the drivers, we need to extract the knowledge from their experience and later reuse it for optimization and logistics issues. This knowledge is given by the files provided by the company that contain, among others, the following fields:

- *Delivery information*. Month, day, pick up and deliver time.
- *Address information*. Postal code, pick up and deliver address.
- *Comments or issues*. Any incidence or comment generated in the delivery process, for example, delays originated by the reception people, comments about the correct place or how to make de delivery in a particular place.

This information has to be loaded into the database. Then, the database contains the driver's knowledge and can be used to, for instance, estimate how much time is necessary to complete a particular ship; or it can be used to indirectly learn from experienced drivers. The drivers know which areas have thick traffic, and other useful information, so they can take alternative paths to reduce time and gas consumption.

During the working day, the driver may stop in places not related to the delivery for different reasons, for instance, to rest or to eat. This information could be extracted from the path variations in specific hours, and although it can theoretically build solutions from a distance perspective, it can be better in time and waste other resources (i.e. gas). However, this information has also some disadvantages, mainly caused by the deficiency and irregularity of the data given by the company. The information has a lot of noise: there are many mistakes in the zip codes and the street names, which is a problem difficult to correct. Even the delivery time, that could be used to calculate the time between some points in the path, is not reliable since sometimes the drivers annotate the hour after they have done some deliveries. For this reason this information has be cleaned.

Then, the following step is to group the deliveries in working areas. We define the concept of *working zone* or *working area* (WA) as a region in a city where one or several drivers work delivering objects (as mentioned before, we have called them *acts*). In other words, a WA is a region that concentrates a high number of acts. For this task we propose *Cased Based Reasoning* (CBR) techniques [1].

Usually, logistic companies define these WA and assign them to a set of drivers using a human expert. Minimizing the overlapping of these WAs among different drivers is essential to minimize the number of drivers and the deliver time. The automatic generation of these WAs is the target of our CBR algorithm. Then, these WAs are used later in the optimization process, i.e., the input of the optimization process is the output of the CBR. The WA that has been created by the CBR, in the context of this work, is named *learning working area*, or LWA.

The generation of the final processed and cleaned information, such as the working areas of the drivers, are carried out using statistical considerations (average of behaviours followed by all the drivers analysed) to minimize the noise.

Finally, each LWA learned is stored as a new case in the database, for each driver, the algorithm is applied and the LWA zones are generated. For more details about this algorithm the reader can refer to [10].

5 Route Optimization

One of the main features of the proposed system is its capacity to suggest efficient routes. The term efficient in this context should be interpreted as relative to a set of "good templates" extracted by the CBR module and provided to the system as input. This input is supposed to have been provided by a human expert. So, the goal of the described system is to improve a given set of human made routes rather than generating complete new routes. This characteristic is used by the optimization engine to guide its search.

The definition of the zones is provided by the CBR described above, so it does not have to handle this task. For this reason, the route optimization can be considered as a variation of the Travel Salesman Problem (TSP). The selection of the optimization algorithm is critical to the success of the system. In any case, the TSP is a well known problem and there are several tools that can be used. One of them is Evolutionary Computation (EC) [2], which provides a set of algorithms that have been successfully used in this scenario, in particular, *Genetic Algorithms* (GA).

GA is a stochastic search algorithm inspired by the biological evolution [4]. As other evolutionary algorithms, it uses a set of candidate solutions called population. In GA, each candidate solution is named chromosome or individual, and, as its name suggests, these chromosomes imitates biological chromosomes. They usually are a string of binary numbers that codify the solution. In order to evolve the population, GA introduces a selective pressure that imitates natural selection. Then, those fittest chromosomes are selected with some controlled randomness using a fitness function, that quantifies the quality of the chromosome. Finally, they are modified by introducing some random mutation and/or mixing two chromosomes in a process named crossover, or recombination.

The design of the GA we propose for route optimization is pretty simple. The chromosome represents the route, just a sequence of integers, each one representing an act. It is, essentially, a combinatory optimization problem. The fitness evaluation is done comparing the solution to the reference route. Of course, a route can outperform or not another route depending on the criteria that is applied. Our system uses four criteria or *qualifications*: Time, distance, zone and act. Using these qualifications we can evaluate different aspects related to the route, such as time or distance.

In order to provide a synthetic estimation of the quality of the individual, these qualifications are aggregated in a linear combination that conforms the fitness function. Each quantification is scalled by a factor, these factors are used to control the weight that the user wants to provide to each quantification, depending on his preferences.

The TSP is a problem where it is difficult, if not impossible, to know when a global maximum is achieved, because of the rapid grown of the search space with the number of acts. So, a convergence criteria is required to stop the execution of the GA. In this case, it is suppossed to have convergence if the average fitness in generation i is less than 1% better than the fitness in generation $i-1$.

In summary, the steps that we have followed in the GA are:

1. *Set the initial population*. A population is generated using the “good templates” extracted by the CBR module, in other words, the initial population is feed with solutions given by human experts. No repetitions of the same individual are allowed. We need to specify how many individuals will be part of the population (this value is set by the user).
2. *Run the algorithm*. Once the initial population is created, the following steps will be repeated.
3. *Selection process*. The fittest individuals (that is, the best fitness values) are selected. The number of individuals that will be saved in the next generation is given by an input parameter.

4. *Crossover*. To generate the rest of individuals we use random points in the actual individuals until we complete the initial population.
5. Results. After those iterations, the *n* first planning routes are presented to the user.

6 Experimental Results

One of the main problems we have encountered when testing, was the high number of irregularities found in the input data provided by the human experts. For this reason the input data had to be preprocessed before it was introduced in the CBR subsystem.

Without any preprocessing, the number of errors in the streets or zip codes is around 45%. It drops to 20% after some light preprocessing (i.e. removing special characters, acute words, etc.), such as using the address to update incorrect zip codes, or removing or adding some characters and searching again. This is still very high when attempting to perform an automatic comparison of the routes followed by the drivers and the ones generated by our tool. So, applying the RL algorithms described in section 3, when preprocessing has been applied, the number of streets that have to be manually corrected is 5%. Figure 2 summarizes these results.

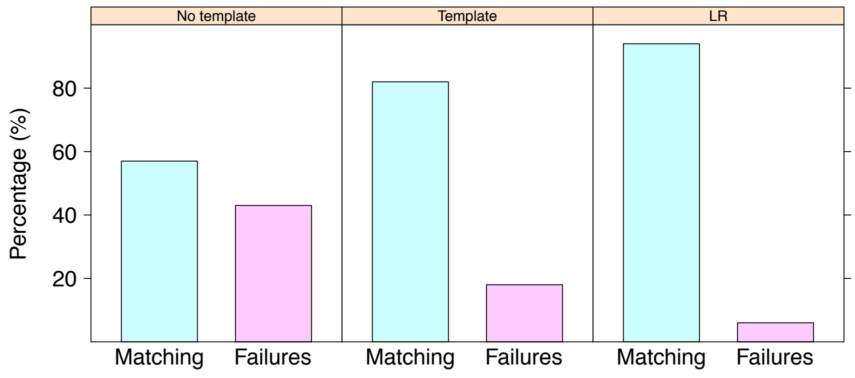


Fig. 2 Effects of three different data cleaning techniques. The percentage of correct and incorrect records in the data is shown.

It is interesting to empirically compare in more detail the RL algorithms described in section 3. Block Distance, Cosine and Dice algorithms get 52% of matches, which is not a notable result. MongeElkan algorithm is even worse, with only 40%. Therefore the only algorithms whose performance makes them suitable are JaroWinkler and Levensthein. In particular, their matches are 90% and 95% respectively. A summary of these results is depicted in Figure 3.

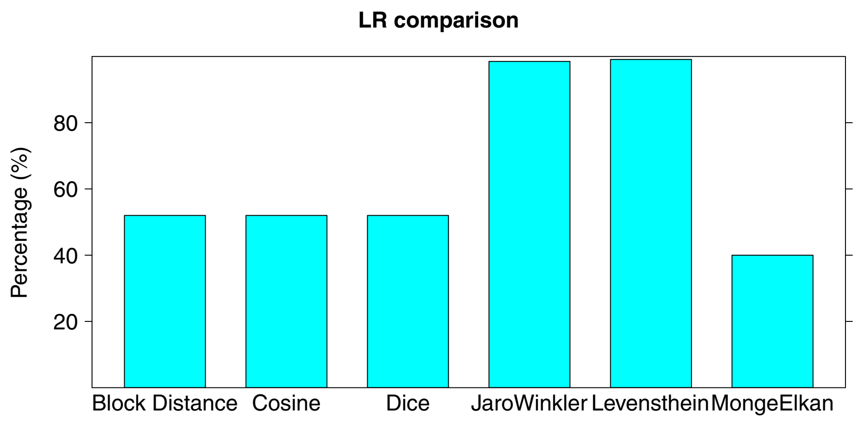


Fig. 3 Comparison of RL algorithms. The graph represents the percentage of matches.

7 Conclusions

In this paper we have presented an AI-based application developed for a Spanish logistics operator company. The main problems the company was interesting in solving are: reusing the experienced drivers knowledge about the route they follow every day and defining routes optimization criteria such as time or fuel.

From the daily interaction between human drivers and the PDAs they use to annotate the deliveries, two main issues can be learned: the particular route that the drivers followed and the addresses. But when dealing with these data, we have detected many mistakes in the addresses introduced so it avoids the system to correctly reuse the available knowledge.

To clean the mistakes on the data and learn the right drivers behaviours, our tool combines Record Linkage (RL), Case-Based Reasoning (CBR) and Evolutionary Computation (EC) techniques. Some of our initial results related to integration of CBR and EC-based plans were presented at [10]. This paper shows how our previous results can be improved using RL techniques. RL allows us to automatically correct the zip codes and the name of the streets that have been badly introduced. CBR is used to separate and learn the most frequent areas that the experienced drivers follow. These techniques allow one to separate the daily incidents that generate noise in the routes, from the decision made based on the knowledge of the route. The EC techniques plan optimal routes from the learning areas and evaluate those routes.

The tool has been tested using real data provided by the company. The results show that we can automate 95% of the data errors, leaving just 5% to the logistics users.

Acknowledgements. We want to thank Antonio Montoya and Antonio Zamora for their contributions in the tool developed. This work has been supported by the Espi & Le Barbier company and the public projects funded by the Spanish Ministry of Science and Innovation ABANT (TIN2010-19872) and by Castilla-La Mancha project PEII11-0079-8929.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications 7(1), 39–52 (1994)
2. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2009)
3. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association (1969)
4. Holland, J.H. (ed.): Adaptation in natural and artificial systems. MIT Press, Cambridge (1992)
5. Jaro, M.A.: Probabilistic linkage of large public health data files. Statistics in Medicine 14(5-7), 491–498 (1995)
6. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady (1966)
7. Monge, A., Elkan, C.: An efficient domain-independent algorithm for detecting approximately duplicate database records. In: Proceedings of the SIGMOD Workshop Data Mining and Knowledge Discovery, pp. 267–270. ACM, New York (1997)
8. Newcombe, H., Kennedy, J., Axford, S.J., James, A.P.: Automatic linkage of vital records. Science (1959)
9. Le Pape, C.: Implementation of resource constraints in ilog schedule: A library for the development of constraint-based scheduling systems. Intelligent Systems Engineering 3, 55–66 (1994)
10. R-Moreno, M.D., Camacho, D., Barrero, D.F., Gutierrez, M.: A decision support system for logistics operations. In: Corchado, E., Novais, P., Analide, C., Sedano, J. (eds.) SOCO 2010. AISC, vol. 73, pp. 103–110. Springer, Heidelberg (2010)
11. Sanders, P., Schultes, D.: Engineering fast route planning algorithms. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 23–36. Springer, Heidelberg (2007)
12. Szczerba, R.J., Galkowski, P., Glickstein, I.S., Ternullo, N.: Robust algorithm for real-time route planning. IEEE Transactions on Aerospace and Electronics Systems 36, 869–878 (2000)
13. Winkler, W.: The state of record linkage and current research problems. Statistics of Income Division, Internal Revenue Service (1999)