Evaluation of Records Similarity in a Duplication Search Engine using Neural Network

Plamen Paskalev, Ventsislav Nikolov

Abstract: *The article discusses the fine tuning and adjustment of a module for detection of duplications, which has been developed as a part of a larger online system. The problems and the current solution are discussed. In this paper a methodology to determine the influence of field similarities on the record pair similarity using a neural network is proposed.*

Index Terms: *artificial intelligence, duplications detection, intelligent user interfaces, neural networks, record linkage*

I. Introduction

Identifying of equality between different descriptions of the same object is quite a common problem in the modern IT industry. Large databases, containing several records per product, person, event or other object, which differ in abbreviations, contain misspellings, typographical errors, non-unique and nonstandard representations of the same logical entity, search engines, data cleaning software, intelligent user interfaces, trying to 'predict' the user's input, thus minimizing the amount of the data to be inputted, are only few areas, this problem takes place. It prevents the proper functioning of the data mining algorithms, collecting statistic data, report features, etc.

There exist various solutions for identifying the duplicated rows, with different level of automatization. In most cases the potentially identical records are provided to the user for confirmation. One important problem for these solutions is the adjustment of the search parameters: they are essential for proper functioning of the algorithm, affect both performance and quality of the results. Configuring of the systems could be confusing and requiring high level of expert competency. An attempt of using neural network for automated calibrating of the search engine is discussed in this paper. In Section II the problem is defined as well as the existing solution. In Section III the suggested solution is discussed in more details. Section IV contains results of applying the described approach, advantages and discussion. Section V contains the conclusions and plans for future improvement of the model.

II. Problem Definition

A realization of a general-purpose duplication search engine was introduced in [1]. The engine receives one record of data and seeks records in a database, which are similar / identical to the investigated one. For comparison a coefficient, representing the level of similarity between two strings as well as between two records (sets of strings with different meaning) are evaluated. Below a brief description of the system is given.

- 1) *Duplication Search engine*

The duplication search engine was developed [1] to help in determination of the level of similarity between two records of data (fig.1).

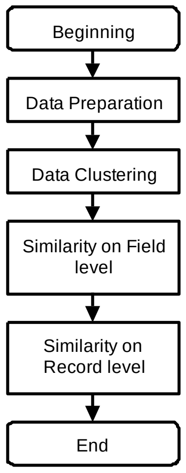

Fig. 1. Duplication Search Engine stages

Multiple different keys are used to determine the clusters of potentially similar rows to be investigated more closely and the results of those different clusters are combined in similar way to the described in [5]. The engine uses edit distance algorithm [2] as a method for determination of the level of likeness between the string fields. It is based on the Levenshtein distance [3], defined as the minimum number of insertions, deletions or substitutions necessary to transform one string into another and Needleman and Wunsch [4] extension, allowing contiguous sequences of mismatched characters, or gaps, in the alignment of two strings. The comparison algorithm implemented by Reinhard Schneider and Chris Sander [6] for comparison of protein sequences, but implemented to compare two ASCII strings is used in the discussed engine. It was extended with including of several features and assignment of weights and bonuses as long as introducing of similarity tables (phonetic similarity, characters, located near to each other on the keyboard, etc.). The engine works with language-dependent interpretation of special symbols (ć š ä ö ß ž, etc.), abbreviations, addresses, fields, containing several words (for example 'Delphi automobile system' against 'automobile systems Delphi'). The engine is realized as a set of modules, written in Java, C / C++, CLIPS. The edit-distance algorithm was realized entirely in

C, the database interface was written in Java (using JDBC, Oracle 10), the duplication search engine (management, analysis of data records, assigning of penalties, parsing of addresses, dates, algorithms for comparison etc.) was realized as a set of production rules for CLIPS. This approach, among the other advantages, gives the opportunity for increasing improvement of the engine with adding or modifying the CLIPS rules.

2) Fields Priority Problem

On the next step, the measured similarity coefficients, calculated for every field, are to be combined to produce general coefficient, representing the level of similarity between two records of data. It is clear, that the fields haven't the same priority, they don't bring same amount of information. For example, the field, containing the name of a person is more informative than a field, containing the birth date or city name. When the field-based results are to be combined in order final similarity coefficient to be calculated, they must be weighted depending on the informative value of the corresponding field.

In the standard approach to the record linkage [7], the database records which to be searched for duplicates are represented by a set of attributes. Considering a candidate pair decision, denoted by y, where y can take values from the set {-1,1}. A value of 1 means, that the records in the pair refer to the same entity and a value of -1 means, that the records in the pair refer to different entities.

Let x = (x1, x2, xn) denote a vector of similarity scores between the attributes corresponding to the records in the candidate pair. Then the probability distribution of y given x is defined as follows:

$$f(x)=\hat{I}\,»_0+\sum_{i=1}^{n}\hat{I}»_i\, x_i \qquad (1)$$

$f(x)$ is known as a discriminant function. $\hat{I}»_i$, for 0 <= i <= n, are the parameters of the model. Given these parameters and the attribute similarity vector x, a candidate pair decision y is predicted to be positive (a match) if f(x) > 0 and predicted to be negative (non-match) otherwise. The parameters are usually set by maximum likelihood or maximum conditional likelihood [8].

Using the equation (1), the duplication match definition can be transformed to

$$f(x)=\sum_{i=1}^{n}\hat{I}»_i\, x_i>\hat{I}\,»_0' \qquad (2)$$

which can be interpreted as general similarity coefficient (calculated based on the similarities between the attributes) must be bigger than a certain limit in order to define the two records as duplicates. In the current solution priority levels are assigned to each of the attributes, involved in the investigation process; then the parameters $\hat{I}»_i$ in (2) are defined as

$$\hat{I}»_i=\frac{weight_i}{\sum_{i=1}^{n} weight_i} \qquad (3)$$

where $weight_i=\max(priority)-priority_i$

The direct assigning of priorities is an expert's task. The person, who will be able to set these coefficients, must be familiar in details not only with the general data content, but with the structure of the database as well. This problem becomes even more serious if a general solution is discussed, which is not hard connected to an area of application. A preferred solution would be one, which needs expert knowledge in the discussed area only. In this way, the structure-dependent characteristics will remain encapsulated in the solution. Because of these reasons a new approach has been investigated and developed – using of a neural network for automatic producing the discriminant function $f(x)$ (2).

III. Using a Neural Network

Making the decision between a conventional and a neural computing solution is not always entirely clear. There are problems for which both conventional and neural approaches may be able to provide appropriate solutions. The choice then depends on the resources available and the ultimate goals of the project.

There are three main criteria which need to be applied when deciding whether a given problem lends itself to a neural computing approach:

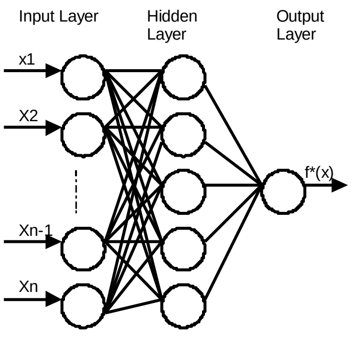

Fig. 2. Input and target output of one training pattern applied to the neural network

- The solution to the problem cannot be explicitly described by an algorithm, a set of equations (representing a physical model, for example), or a set of rules or it is too difficult.
- There is some evidence that an input-output mapping exists between a set of input variables x and corresponding output data y, such that y = f(x). The form of f() , however, is not known. In the described solution the defining of the f() is an expert's task.
- There should be a large amount of data available, i.e. many different examples with which to train the network.

A. *Constructing of Neural Network Architecture*

The number of inputs in a multilayer network is determined by the number of features or input parameters available for the problem under consideration. Thus, the neural network in the described solution has 14 input and one output units (corresponding to the size of input and output data respectively in one learning pattern).

The neural network was written using C# and it has one hidden layer. The activation function used is one of the most typical activation functions [9] – the binary sigmoid function. which has range of (0, 1).

(4)

Initialization of the weights involves set of small random values of (-0.1, 0.1). In addition to weights, bias values to the neurons are involved in computing of network output. The learning rate for each of the patterns is 0.3. The choice of learning rate can have a significant effect on the performance of a network. Well-chosen learning rate will move the weight toward their optimal values in reasonable time. If some training data are very different from the majority of the data (or some part is incorrect), momentum factor should be used. Because the weights are initialized with random values, gradient descent process could last unpredictable time. That's why training continued until reached defined number of epochs.

B. *Experimental Methodology*

Two main elements are specific:

- the collection, preparation and analysis of the training data
- the design, training and testing of the neural network

Two experiments were conducted to determine the research task. The experiments were conducted first for the expert system model and then for the neural network model. Both experiments used the same task (determination of result percentage of record similarity based on given similarity of their fields), the same data set, and the same dependent variables, except the field priority which is determined in expert system model and not known in neural network model. The data assumed as correct was obtained using expert system approach and it was clustered in appropriate input classes for the training set of neural network (input vectors) and results were used as correct output vectors. That data is separated for both training and testing set of patterns.

The neural computing research literature is rich of papers which propose many different network architectures, but they are mainly variations of multilayer perceptron and radial basis function architectures [10]. In the same way, neural network used for the experimental results is standard multilayer perceptron with one hidden layer and backpropagation supervised learning. The network topology is constrained to be feedforward: i.e. loop-free - generally connections are allowed from the input layer to the hidden layer and from the hidden layer to the output layer. The hidden layer learns to recode (or to provide a representation for) the inputs [13].

Applications using neural network approach involve mapping a given set of inputs to a specified set of target outputs. The aim is to achieve network that could give reasonable responses to input that is similar, but not identical to that, used in training. The training involves feed-forward of the input pattern, the calculation of the error and adjustment of the weights and biases. In some cases it is slow, but a trained net produces its output rapidly.

IV. Implementation of a Neural Network Approach

It is important to make a reasonable estimate of how much data is required to train the neural network properly. If too small amount of data is collected, the full range of the relationship that the neural network should be learning may not be covered. The experiment need to have sufficient data points for the form of the mapping to be specified accurately enough throughout the whole range of input space of patterns. If there is no training data for a region of input space from which some of the test data is drawn, then there cannot be any valid generalisation for these patterns. In that case the neural network will always give an answer, but it will not be precise (fig.4). In the test, shown on this figure, the neural network was trained with similarities between 30%-60%. The deviation between the similarity calculated using the existing, priority-based approach and the one, calculated using the neural network, was too high, especially in the areas, outside of this range (If there is no deviation between both approaches the points should lay on the diagonal line). Thus, to ensure that the neural network is trained well to recognize and respond to the full range of values, 270.000 potential training patterns (input-output vectors) separated in several classes were collected using the priority-based approach. Each class contains examples which have output in the range of 10 percent and corresponding to their input. For instance, record similarity of 0 – 9% form class one, 10 – 19% form class two, etc. There were assembled two balanced couples of sets in which all the classes are uniformly represented. The first couple contains 50 learning and 250 testing patterns, the second 100 learning and 250 testing patterns. To start the construction of the neural network 14 variables were identified from each learning pattern as input (each record contains 14 fields). They are specified as in the example below:

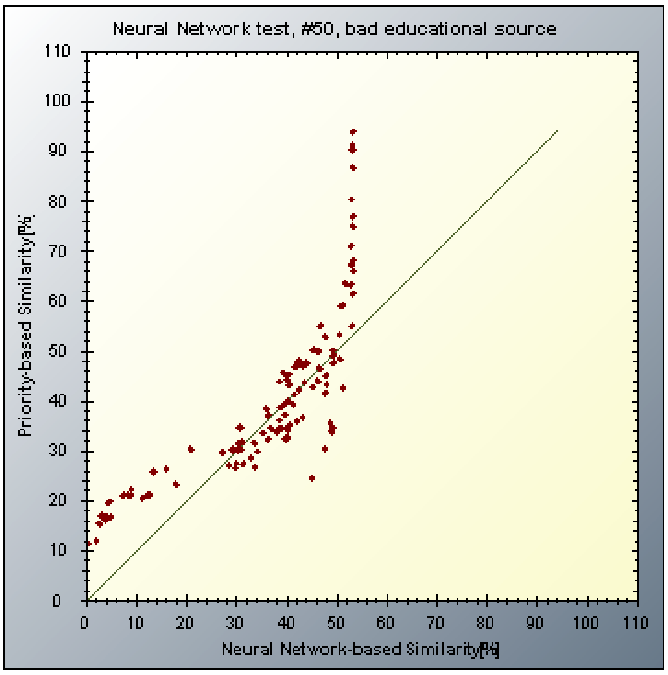

Fig. 3. A test with badly selected training patterns.

1 47.50 2 0.00 3 31.67 5 100.00 6 100.00 7 -3.00 8 25.00 9 0.00 11 0.00 12 0.00 13 0.00 14 0.00

In the above sequence are alternate integer and real data values composing couples of numbers. The integer ones correspond to the subsequent number of the data field participating in the training vector (that means it is the number of input neuron which will accept it), the real - its value which represents the string similarity for that field number. Each one of those 14 input values correspond to the similarity of one field for a record (f.e. first name, last name, city etc.) and it is defined using expert system. If there is a field that don't participate in evaluation, its couple absents. It means that data is missing for this particular field in the database. Then his position in forming input vector contains zero. The example above is converted to the following input of neural network:

47.50 0.00 31.67 0.00 100.00 100.00 -3.00 25.00 0.00 0.00 0.00 0.00 0.00 0.00

Each one of these values is input for one neuron from the input layer of the network. Because of the need to minimize the effect of magnitude, during the pre-processing stage the input values are normalized, $x_i \in (-1,+1)$. The final form of input pattern is:

0.475 0.00 0.3167 0.00 1.0 1.0 -0.03 0.25 0.00 0.00 0.00 0.00 0.00 0.00

The corresponding output of the neural network includes only one value – the estimated value of the records pair similarity, discriminant function $f(x)$ as defined in (2).

0.475 0.00 0.3167 0.00 0.1 0.1 -0.03 0.25 0.00 0.00 0.00 0.00 0.00 0.00 – input vector

0.2447 - output vector

Test data is in the same form. It was also selected uniformly from the classes, discussed above in the same way as educational data. These two sets are random and don't overlap each other. Each input vector of the testing examples was passed through the network which produce output vector (of one value). Then the results of the two approaches were juxtaposed to be summarized and graphically represented in acceptable form.

There is several known rules for the number of units in the hidden layer. If the number of units is too few it will result in underfitting. If there are too many units it can result in overfitting and increased time for training the network. In our case that number is set using trial and error approach. As a starting point the number of hidden neurons was choosen to be less than twice the input layer size.

There were several tests performed until a suitable combination of number of training patterns and other parameters was found.

The test with 50 training patterns, uniformly distributed between 0% .. 100% , have shown good representation in the lower part (up to 50% similarity), but with higher deviation in the higher part of the chart (fig.5). The test (fig.5) was performed with the following conditions:

- Number of training patterns: 50
- Number of test patterns: 250
- Number of input neurons: 14
- Number of hidden layers: 1
- Number of neurons in the hidden layer:20
- Number of output neurons:1
- Number of training patterns: 50
- Learning rate: 0.3
- Momentum factor: 0.0

Achieved average squared error: 0.00469551606559766

The test with 100 training patterns, uniformly distributed between 0% .. 100% , have shown very good distribution compared to the priority-based cases along the all values of the test portfolio (fig.6).

The test was performed under the following conditions:

- Number of training patterns: 100
- Number of test patterns: 250
- Number of input neurons: 14
- Number of hidden layers: 1

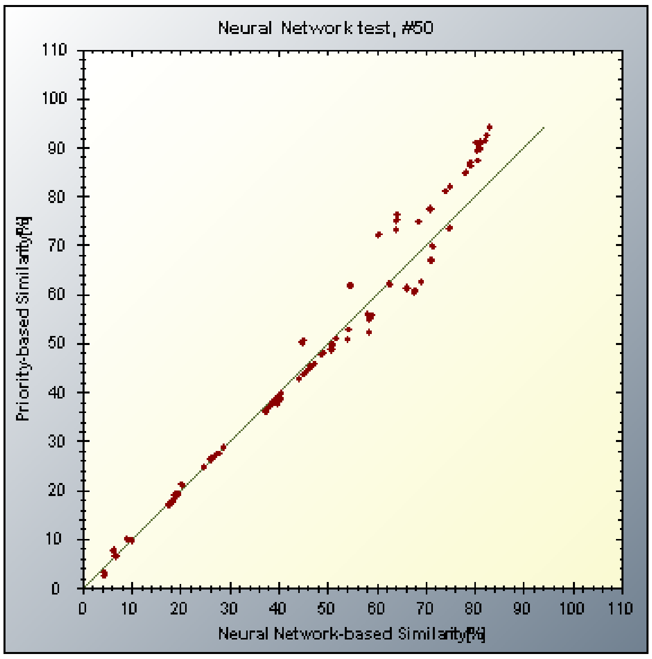

Fig. 4. A test with 50 training patterns

- Number of neurons in the hidden layer: 10
- Number of output neurons:1
- Learning rate: 0.3
- Momentum factor: 0.0
- Achieved average squared error: 0.00432612246581036

This solution was accepted as accurate enough for using in the application.

V. Conclusions and Future Works

- One direction for further improvement of the discussed solution is in additional tests with the neural network. The neural network could use more effective procedures for weights initialization like Nguyen-Widrow initialization, optimization about hidden layer(s) or activation functions, appropriate for given data, different techniques to determine dynamically during the training the values of the learning rate and momentum factor, using algorithms for improve the performance of basic gradient descent.

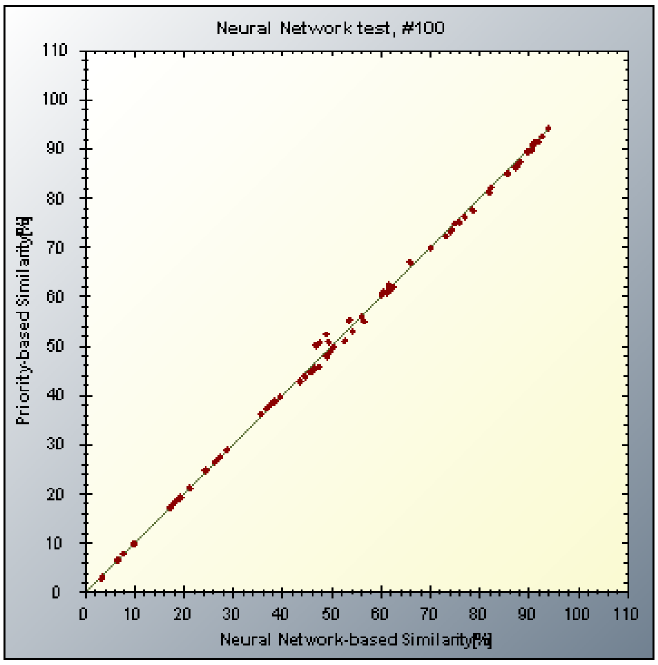

Fig. 5. A test with 100 training patterns

- The setting for various parameters (number of hidden neurons, number of learning epochs, number and distribution of training data patterns etc.) is a major challenge. Some of these parameters was chosen by the principle of trial and error. The developed network does not always give optimal solution, but with appropriate settings improvement it can produce better answer.
- The measure of system's performance in test data is generally expressed in terms of accuracy, correct output (based on comparison with expert system results) and error rate. Based on preliminary well selected patterns neural network approach can give quite accurate results of record similarity compared to expert system approach.
- The duplication search engine is used in two independent ways (fig. 7):

  - As a part of a WEB-based application for research and analyzing of large data bases, containing information for customers. The database in this case contains duplicates, which must be found and connected to common objects of a superstructure, thus providing a good platform for optimization, data clearing and analyzes. In this case the duplication search engine works as part of Business Layer in a mode, when the accuracy of the returned results is extremely important.
  - As a part of an Intelligent User Interface (IUI) module. In this case the main task of the engine is, via searching both own database and the database of the application, to feed the IUI with records which match the information, the user is currently entering in the GUI. In this way the IUI, using own rule-based mechanism, is trying to predict the users enter and to simplify his tasks. In this case, although the accuracy is also important, the time for reaction is essential. The response must have place in real time in order to give chance to IUI to produce its own output.

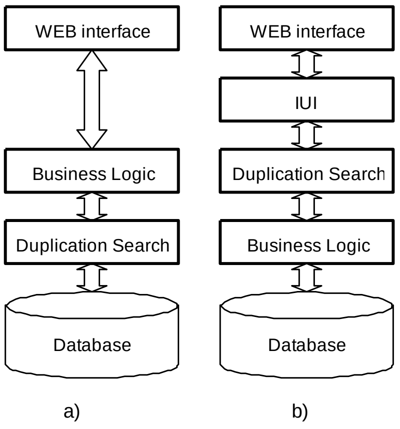

Fig. 7. Using of Duplicate Search Engine.

Although the results using the Neural Networks, described in this article, match requirements for both cases quite well, additional profiled investigation for both specific cases must be performed.

References

[1] P. Paskalev, A. Antonov, “Intelligent application for duplication detection”. In proceedings of the *International Conference CompSysTech 2006*, IIIA.27.1-IIIA.27.8, June. 2006.

[2] Gusfield “*Algorithms on strings, trees and sequences*”. Cambridge Univ. Press, NY, 1997

[3] Bilenko M., Mooney R. “Adaptive duplicate detection using learnable string similarity measures” In Proceedings of the *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*(KDD-2003), Washington DC, pp.39-48, August, 2003

[4] Needleman S. B. and Wunsch C. D. “A general method applicable to the search for similarities in the amino acid sequences of two proteins.” *Journal of Molecular Biology*, 48:443–453, 1970

[5] Hernandez M. A. and Stolfo S. J. “The merge/purge problem for large databases.” In Proceedings of the *1995 ACM SIGMOD*, pages 127–138, San Jose, CA, May 1995.

[6] Sander C. and Schneider R., "Database of homology-derived protein structures and the structural meaning of sequence alignment," *Proteins*, vol. 9, no. 1, pp. 56--58, 1991.

[7] I. Fellegi and A. Sunter. “A theory for record linkage*”. Journal of the American Statistical Association*, 64:1183-1210, 1969

[8] Parag and Pedro Domingos. ”Multi-relational record linkage”. KDD-2004 *Workshop on Multi-Relational Data Mining* (pp. 31-48), 2004

[9] Laurene Fausett, “*Fundamentals of neural networks. architectures, algorithms and applications*”. Prentice Hall, 1994, ISBN: 0133341860

[10] M. Young, “*The Technical Writer's Handbook*”. Mill Valley, CA: University Science, 1989.

[11] Lionel Tarassenko, “*A Guide to Neural Computing Applications*”. Butterworth-Heinemann, 1998, ISBN: 0340705892

[12] Barbara D. Klein and Donald F. Rossin “Data errors in neural network and linear regression models: An experimental comparison” Data Quality, vol5, n1, 1999– www.dataquality.com/999KR.htm

[13] Leslie Smith, “*An Introduction to Neural Networks*”, http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html