How to Design a Network of Comparators

Łukasz Sosnowski[1,2] and Dominik Ślęzak[3,4]

[1] Systems Research Institute, Polish Academy of Sciences
ul. Newelska 6, 01-447 Warsaw, Poland
[2] Dituel Sp. z o.o.
ul. Ostrobramska 101 lok. 206, 04-041 Warsaw, Poland
[3] Institute of Mathematics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
[4] Infobright Inc.
ul. Krzywickiego 34 lok. 219, 02-078 Warsaw, Poland
l.sosnowski@dituel.pl, slezak@infobright.com

Abstract. We discuss the networks of comparators designed for the task of compound object identification. We show how to process input objects by means of their ontology-based attribute representations through the layers of hierarchical structure in order to assembly the degrees of their resemblance to objects in the reference set. We present some examples illustrating how to use the networks of comparators in the areas of image recognition and text processing. We also investigate the ability of the networks of comparators to scale with respect to various aspects of complexity of considered compound object identification problems.

Keywords: Comparators, Compound objects, Hierarchical models.

1 Introduction

Decision making is one of the key aspects of our lives. Although the decision-making processes may seem to be simple and intuitive for humans, their automatic support raises a number of challenges related to compoundness of instances that we want to reason about, as well as computational complexity of the corresponding data exploration procedures. Consequently, the designers of decision support systems usually attempt to decompose the corresponding tasks onto their more basic components, often organizing them into multi-layered structures. Such structures may follow extensions of neural networks [1,2] or hierarchical learning models [3,4]. This approach allows for building and using complex solutions step by step, using relatively simple computational units.

Analogous strategy can be used for solutions requiring comparisons of compound objects, which underly a wide class of approaches to classification, prediction, search and others. In our previous research in this area [5], we focused on applications of so called fuzzy comparators in compound object identification. In [6], we noted that the network-like comparator structures could be useful for decreasing the amount and complexity of necessary comparisons. In [7], we discussed how to employ some massive data processing methodologies to better organize comparison operations during object identification process.

K. Imamura et al. (Eds.): BHI 2013, LNAI 8211, pp. 389–398, 2013.
© Springer International Publishing Switzerland 2013

The purpose of this paper is to focus on practical applications of comparator networks. We show that complex decision-making solutions can be modelled by units with relatively basic functionality of expressing and synthesizing the degrees of resemblance, proximity or similarity between objects, their sub-objects and their structural representations. We also emphasize simplicity of proposed network models, with their layers corresponding to some domain ontology classes [8], their nodes corresponding to appropriately chosen objects' attributes [9], and the sets of so called reference objects which can be interpreted using the terminology of instance selection in knowledge discovery [10].

The paper is organized as follows. Section 2 introduces the most basic concepts related to our way of looking at the problem of comparing compound objects. Section 3 discusses the principles of constructing comparator networks for the purposes of object identification. Sections 4 and 5 illustrate those principles by means of practical examples from the areas of image recognition and text processing, respectively. Section 6 concludes our work.

2 Comparisons of Compound Objects

The data processing systems often need to deal with compound objects represented in various forms. If our task is to identify the input objects by comparing them to elements of some reference set, we should understand their structures, relations and properties in order to build an appropriate comparison model. Among many challenges we can list here a usual lack of precision of object representations and instability of attributes that describe them. This is why there is a need for reasoning about objects by referring to the degrees of their pairwise resemblance, instead of judging them as identical or different.

The way of expressing the knowledge about the input object is usually closely related to an ontology specific for the real-world domain of the analyzed objects [11]. The ontology models concepts and attributes describing objects, as well as dependencies and relationships with other objects. An important requirement is that object descriptions are sufficiently complete. For the task of identification, the domain ontology should let derive enough concepts to clearly distinguish between the entities using relationships in the given field. One can consider a kind of reduct – a minimal subset of attributes able to span a layer of comparison units which all together allow for valid object identification.

Apart from appropriate selection of attributes, the identification model should include a set of reference objects which the input cases will be compared to. Depending on the specific area of application, the set of reference objects can be fixed, or it may evolve with inflow of information to the model. Selection of attributes and reference objects is equally important for the domain experts in order to understand and influence the identification process. The reference objects should be representative for all important aspects of a given identification task. Further ideas how to simultaneously optimize the sets of attributes and reference objects can be found e.g. in [12], where so called information bireducts are employed to learn similarity models from textual data.

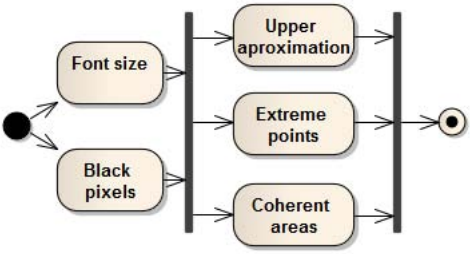

Fig. 1. The network of comparators for the OCR task considered in Section 3

Once the attributes and reference sets are established, we can adapt a variety of approaches to compute object resemblances. Such approaches are often based on deriving the degrees of membership, proximity or similarity of the considered input objects to some templates or representatives [13,14]. For structural objects, it is important to take an advantage of the knowledge how they can be meaningfully decomposed onto sub-objects. The computation can be then conducted at the level of simpler comparisons. The partial outcomes of such comparisons can be further combined by specific rules. The input objects and their sub-objects can be labeled with multiple proposed templates. The attributes of the most relevant templates can be then assigned to the considered input objects.

3 Construction of Comparator Networks

A comparator network is constructed by putting together basic components: comparators, aggregators and translators [6]. The network is organized in layers that follow from the domain ontology used to describe objects. Each input object is processed through the whole network. The nodes in every layer correspond to comparators. A comparator is a computational unit which produces a subset of reference objects that the given input resembles in the highest degree with respect to a single attribute. Subsets of reference objects are aggregated in order to better direct the comparisons to be performed in further layers. Attributes examined in the nodes of a single layer can be usually computed independently, so the corresponding comparisons are conducted concurrently [7].

The meaning of reference objects may remain the same or vary across the layers. In the first case, further referred as homogeneous, the comparator outputs are usually aggregated by (possibly ranked or weighted) intersection operations over reference subsets. In the second, heterogeneous case, different layers can analyze the same input object at different levels of its structure. For example, given a floor plan as an input, we can start by comparing its components (doors, corridors, etc.) to the corresponding sub-objects of reference floor plans. We can continue with higher-level comparisons after translating the outputs of the first layer into reference objects at the level appropriate for next layers.

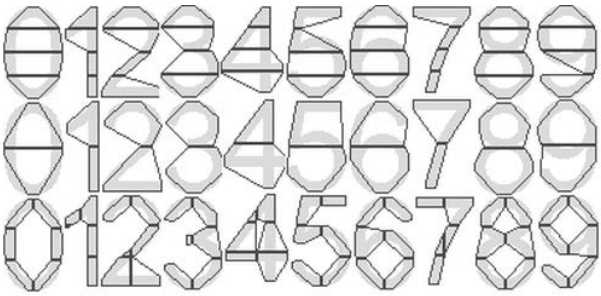

Fig. 2. Reference digits prepared for comparisons with respect to their shapes. (With granulation resolution parameters 3×3, 2×5, 2×2 displayed in consecutive rows.)

During translation of the collections of (ranked or weighted) reference objects between layers, we need to deal with the two following problems: First of all, the resemblance weights computed in the previous layer at the level of reference sub-objects (or super-objects) need to be translated to resemblances of reference objects specific for the next layer. Secondly, during such translation, there is a need of applying properly designed rules which prohibit combining sub-objects in an invalid way. Such exclusion rules turn out to be useful also in the internals of lower-level comparators. This is because some aspects of object comparisons cannot be expressed by typically used fuzzy similarity measures [5].

To sum up, in order to design an intuitive network layout, we should utilize the domain knowledge about the studied objects for the purposes of defining the contexts and attributes responsible for resemblance measurements. We should also establish a reference set for a specific identification problem. Then, for each input object, we can observe its processing through the network layers, until the final aggregator gives us the most relevant element (or elements) of the reference set. The detailed mathematical specifications of comparators, aggregators and translators are important for the network's efficiency as well, although they may not be so easy to understand for the domain experts. Appropriate algorithms should be employed to learn and tune both parameters of the network units and parameters responsible for creating and updating reference objects.

4 Case Study: Identification of License Plates

This section serves as an introductory illustration of our approach. Let us start with a network shown in Figure 1, designed for a simplified character recognition problem. We assume that the input strings of digits are segmented onto single elements. Further, we convert each of digits (including the reference set elements) into to the binary scale and cut the result in such a way that each edge of the newly created image is one-pixel distant from the font's black edge.

Let us construct a homogeneous network with two layers. The first layer covers easily computable attributes which can significantly reduce the amount of

Table 1. Structural reference set for license plates in Poland. (L/D – letter/digit.)

1st group	2nd group
LL	DDDDD
LL	DDDDL
LL	DDDLL
LL	DLDDD
LL	DLLDD
LLL	DDLD

1st group continued	2nd group continued
LLL	DDLL
LLL	DLDD
LLL	DLLD
LLL	LDDD
LLL	LDDL

1st group continued	2nd group continued
LLL	LDLL
LLL	LLDD
LLL	DDDDD
LLL	DDDDL
LLL	DDDLL

reference objects potentially relevant to each given input. The second layer relates to an in-depth image analysis leading to further narrowed down reference subsets finally synthesized by the output aggregator. The reference set consists of images of considered digits grouped by different sizes and font types. Thus, the cardinality of the reference set significantly exceeds the amount of distinct digits. One may wish to look at it as an illustration of a potential application of comparator networks to the classification problems, where different decision classes are represented by groups of examples in the same reference set.

Figure 2 illustrates the choice of comparators. The first layer analyzes the size of a font and the occurrence of pixels in the image. For the given input digit, both comparators produce the weights corresponding to its size and pixel related resemblance to some subsets of reference images. These weighted subsets are intersected (with some tolerance threshold) by the aggregator and passed to the second layer as a new reference set. The next three comparators are slightly more advanced (see [5] for more details). The first of them compares the upper approximations of original images by means of granules corresponding to very low resolution parameters ($m \times n$). The second comparator focuses on geometric shapes created by connecting the extreme points in particular granules. The third comparator looks at coherent areas within the analyzed objects, which usually allows for distinguishing between digits 0,4,6,8,9 and 1,2,3,5,7.

If the aggregator is implemented as an intersection with some tolerance threshold, then a reference object can be passed to the next layer even if it does not match with the input digit on all attributes. The parameters of such aggregation should be adjusted to achieve an acceptable tradeoff between minimizing cardinalities of the produced reference subsets and avoiding elimination of the eventually best-matching reference object. On the other hand, for some borderline cases, a larger weighted subset of reference objects corresponding to different digits should be returned. In general, the parameters of comparators or aggregators need to be tuned by applying some (semi-)supervised learning algorithms or an additional interaction with the domain experts [4].

Let us now show how to employ the above implementations in the task of license plate identification. We consider an example of the license plates in Poland, which have a clear structure: their first parts consist of two or three letters (depending on the type of a county) while their second parts contain both letters and digits. The first license plate segment can fully evidenced in the set of reference sub-objects. The second part contains five (11) possible schemes under the

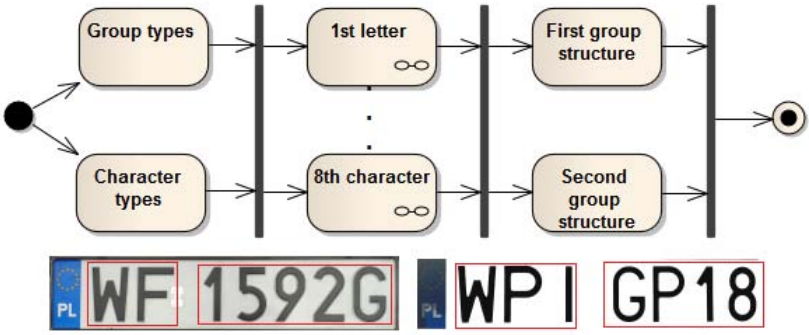

Fig. 3. The network for identification of license plates in Poland

assumption that the first part consists of two (three) letters, as reported in Table 1. Such structural knowledge should be taken into account while constructing the solution. This way of representing and utilizing the domain knowledge about data content shares also important analogies with other approaches [8,15].

Figure 3 illustrates the proposed heterogeneous network. Let us assume that the input objects are already preprocessed, i.e., they are decomposed onto images of single characters. The input layer of our network determines the structural type of the incoming license plate (2-5, 3-4, or 3-5) and the type for each of its characters (digit or letter). These initial results are stored in the information granule built around the input object and passed to homogeneous sub-networks analogous to the one in Figure 1, each of which is responsible for identifying an individual character. The next layer receives the sub-network outcomes in a form of weighted subsets of digits/letters recommended for particular positions on the license plate. Such outcomes are combined in order to obtain a weighted set of possible license plates. The translation from the level of single character sub-objects to license plate objects requires combining the partial weights (or rankings) and taking into account the constraints defined by the structural reference set in Table 1. The design of this stage needs to carefully follow the principles of hierarchical modeling of similarities between compound objects [11,14].

5 Case Study: Semantic Parsing of Scientific Publications

This section refers to an academic project in the area of semantic search over a repository of scientific publications [16]. One of the requirements for the project's success was to assure good quality of the repository's content. For this purpose, we designed a data warehouse with two layers: 1) the instance layer storing generic information acquired from external scientific content repositories, and 2) the object layer storing cleaned and completed information about properties and relations between objects such as scientists, (parts of) publications, their topics and others. The object layer provides an input to the intelligent methods for semantic indexing and similarity learning (see e.g. [12]). The instance layer

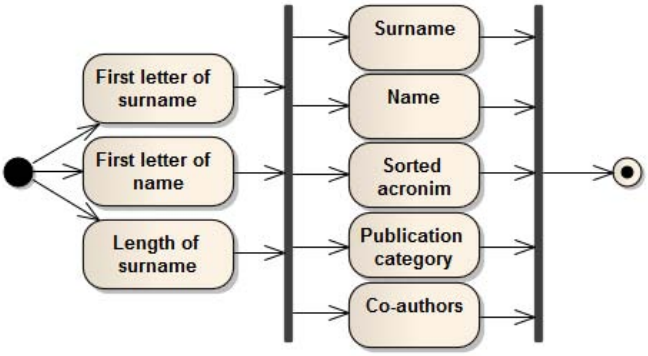

Fig. 4. The network for identification of publication authors

allows for better access to the original data and provides necessary background for the advanced object deduplication and integration algorithms which cannot be handled in real time, while loading new data into the repository.

Let us discuss two specific tasks for which comparator networks turn out to be useful. In both cases, the above-mentioned object layer will serve as the reference set for the analysis of a fresh content of the instance layer.

The first example refers to the problem of identifying scientists. For a number of scientific repositories, publications are annotated with their authors' names. However, there is no straightforward method to decide whether different publications are (co-)authored by the same person. There may be different scientists with the same names. On the other hand, the same scientist may be represented in different ways (with initials or full names, with mistakes etc.). Thus, our task is to maintain the reference set of already identified scientists and to evaluate whether the authors of newly loaded publications belong to this set.

Figure 4 displays the proposed homogeneous network. The input layer consists of three concurrently running comparators focused on the following attributes: the surname first letter, the name first letter, and the surname length. The first two comparators work in a strict 0/1 mode. The third one may allow for some flexibility, e.g., a difference up to one character). All these comparators are computationally very simple. Therefore, for each newly considered instance of an author's name, the subset of already stored potentially relevant scientists will be quickly narrowed down by aggregating comparator outcomes.

The second layer utilizes five further comparators investigating the resemblance of the input instances to the elements of the above-obtained reference subsets by means of the name, the surname, the sorted acronym, the publication category and the set of co-authors. The three first attribtues refer directly to the string representations of scientists in the repository. The forth attribute refers to the relation of the input author instance with a scientific category retrieved for the newly loaded publication. It is then compared with categories of already stored publications of reference objects. The fifth attribute is managed

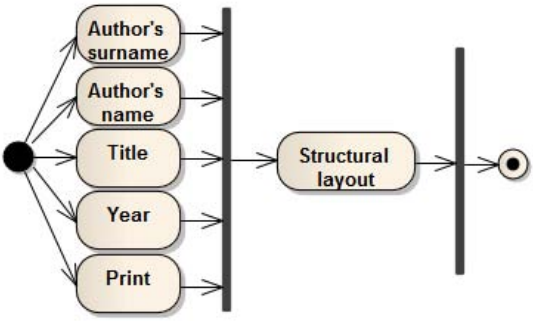

Fig. 5. The network for semantic parsing of publication information

analogously. One can see that the last two comparators may need to base on quite incomplete information. Moreover, their work is more computationally intensive than for the previous ones. This is why they are placed in the second layer, where the subsets of relevant reference objects are already limited.

There are several possibilities of employing the final network's outcome. If the newly loaded author's name instance does not sufficiently resemble any of already stored scientists, the new entry should be created in the repository's object layer. If the input resembles a single existing object, it should be utilized to additionally verify and/or complete information about the properties and relations of that object. Also, a link between the new instance and the object should be recorded in the repository. If the input resembles multiple objects, a procedure of merging them into a new single object may be triggered. If the input strongly resembles an already existing object only for a subset of comparators, a split of already stored instances linked to that object onto some subsets corresponding to multiple new objects may be required. Obviously, it is then crucial to tune parameters responsible for choosing among all above options in order to achieve optimal quality and granularity of objects, their properties and their relations.

The remaining part of this section refers to publication descriptions for which it is difficult to extract components corresponding to authors, titles and other properties. This situation can be often observed for the bibliography items occurring at the ends of publications loaded into the repository. Such items should be interpreted, parsed and loaded as the publications themselves, but the online text analysis algorithms may not handle it credibly enough. In our project [16], we follow the strategy of loading such descriptions into the repository as unparsed strings with no properties and related instances assigned. Then, a specific sub-layer responsible for extracting important information from such strings is applied. The idea is to discover internal structures of the input bibliography descriptions by comparing their dynamically identified components with various types of objects already stored in the repository. Such components can be then treated as new entries in the instance layer, ready for further analysis.

Table 2. Left: A sample of reference structures. Right: An input and its decomposition with degrees of resemblance of its components to the corresponding reference sets.

Abbreviation dictionary	
A - authors	*T* - title
J - journal	*B* - book
Y - year	*Pb* - publisher
P - pages	*S* - series
V - volume	*E* - editors
N - note	...

Reference structural objects
A T B S V Y P
E T B S Pb Y V
A T B Y
A T J Y
A T S Pb Y
...

Example of the input object
M. Szczuka, Ł. Sosnowski, A. Krasuski, K. Kreński, "Using Domain Knowledge in Initial Stages of KDD: Optimization of Compound Object Processing," Fundam. Inform., 2013, to appear.

Comp	Substring	*A*	*T*	*Y*	*P*	*J*
1	M. Szczuka	0.83	0	0	0	0
2	Ł. Sosnowski	0.83	0	0	0	0
3	A. Krasuski	0.85	0	0	0	0
4	K. Kreński	0.84	0	0	0	0
5	Using (...)	0	0.75	0	0	0
6	Fundam. (...)	0	0.42	0	0	0.55
7	2013	0	0	1	0	0
8	to appear	0	0	0	0	0

Let us consider the heterogeneous network in Figure 4. We start by cutting the input text onto pieces, e.g., by means of characters such as dots, commas, etc. [15]. As we do not know which pieces correspond to particular types of the bibliography item components (more formally, properties and relations of the considered publication with some other objects), the first network's layer focuses on computation of resemblance degrees of all pieces to the reference sets maintained in the repository for scientists' surnames, names, initials, publication titles, years and others. Table 2 shows a simplified example of the outcome of the stages of text decomposition and resemblance calculation. As in some cases the assignments of substrings to particular component types are problematic, the identification process is strengthened by the analysis of the bibliography item's structure at the next network layer. Table 2 also displays some examples of reference structural objects, i.e., the structures of bibliography items already stored in the repository. The structural layout comparator in Figure 4 determines the most reasonable hypothesis about the input's structure basing on such reference objects, in combination with information obtained in the previous layer. The final assignment is retrieved by following the most probable structure.

6 Conclusion

We outlined some practical aspects of applying comparator networks in compound object identification. We discussed how to define ontology-based contexts for the network layers, how to choose attributes to be compared in the network units, and how to manage the sets of reference objects. We also emphasized the importance of mechanisms of propagating the degrees of resemblance between the input and reference (sub-)objects through the network.

Acknowledgments. This research was supported by grants SP/I/1/77065/10 ("Interdisciplinary System for Interactive Scientific and Scientific-Technical Information") and O ROB/0010/03/001 ("Modern Engineering Tools for Decision Support for Commanders of the State Fire Service of Poland during Fire & Rescue Operations in the Buildings") founded by Polish National Centre for Research and Development (NCBiR), as well as grants 2011/01/B/ST6/03867 and 2012/05/B/ST6/03215 founded by Polish National Science Centre (NCN).

References

1. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.J.: A Tutorial on Energy-based Learning. In: Predicting Structured Data. Neural Information Processing Systems. MIT Press (2007)
2. Lingras, P.: Fuzzy-Rough and Rough-Fuzzy Serial Combinations in Neurocomputing. Neurocomputing 36(1-4), 29–44 (2001)
3. Bengio, Y.: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
4. Nguyen, S.H., Nguyen, T.T., Szczuka, M., Nguyen, H.S.: An Approach to Pattern Recognition based on Hierarchical Granular Computing. Fundamenta Informaticae 127(1-4), 369–384 (2013)
5. Sosnowski, Ł., Ślęzak, D.: Comparators for Compound Object Identification. In: Kuznetsov, S.O., Ślęzak, D., Hepting, D.H., Mirkin, B.G. (eds.) RSFDGrC 2011. LNCS, vol. 6743, pp. 342–349. Springer, Heidelberg (2011)
6. Sosnowski, Ł., Ślęzak, D.: Networks of Compound Object Comparators. In: Proc. of FUZZ-IEEE 2013 (2013)
7. Ślęzak, D., Sosnowski, Ł.: SQL-based Compound Object Comparators: A Case Study of Images Stored in ICE. In: Kim, T.-H., Kim, H.-K., Khan, M.K., Kiumi, A., Fang, W.-C., Ślęzak, D. (eds.) ASEA 2010. CCIS, vol. 117, pp. 303–316. Springer, Heidelberg (2010)
8. Bazan, J.G., Buregwa-Czuma, S., Jankowski, A.: A Domain Knowledge as a Tool for Improving Classifiers. Fundamenta Informaticae 127(1-4), 495–511 (2013)
9. Świniarski, R.W., Skowron, A.: Rough Set Methods in Feature Selection and Recognition. Pattern Recognition Letters 24(6), 833–849 (2003)
10. Verbiest, N., Cornelis, C., Herrera, F.: FRPS: A Fuzzy Rough Prototype Selection Method. Pattern Recognition 46(10), 2770–2782 (2013)
11. Pawlak, Z., Skowron, A.: Rough Sets: Some Extensions. Information Sciences 177(1), 28–40 (2007)
12. Janusz, A., Ślęzak, D., Nguyen, H.S.: Unsupervised Similarity Learning from Textual Data. Fundamenta Informaticae 119(3-4), 319–336 (2012)
13. Marin, N., Medina, J.M., Pons, O., Sanchez, D., Vila, M.A.: Complex Object Comparison in a Fuzzy Context. Information and Software Technology 45, 431–444 (2003)
14. Polkowski, L., Skowron, A.: Rough Mereological Calculi of Granules: A Rough Set Approach to Computation. Computational Intelligence 17(3), 472–492 (2001)
15. Kowalski, M., Ślęzak, D., Toppin, G., Wojna, A.: Injecting Domain Knowledge into RDBMS – Compression of Alphanumeric Data Attributes. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 386–395. Springer, Heidelberg (2011)
16. Ślęzak, D., Stencel, K., Nguyen, H.S.: (No)SQL Platform for Scalable Semantic Processing of Fast Growing Document Repositories. ERCIM News 90 (2012)