Fuzzy Set Interpretation of Comparator Networks

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
l.sosnowski@dituel.pl
[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
slezak@mimuw.edu.pl

Abstract. We discuss how to model similarities between compound objects by utilizing networks of comparators. The framework is used to construct identification and classification systems. Comparing to our previous research, we pay a special attention to fuzzy-set-inspired foundations of how compound signals are processed through the network. We also reconsider some of already-known examples of applications of comparator networks, now using the proposed fuzzy-set-based terminology.

Keywords: Networks of comparators · Compound object similarities · Fuzzy sets and relations · Semantic parsing of bibliography items

1 Introduction

Similarity is one of fundamental aspects of reasoning in artificial intelligence [1]. In this paper, we show a similarity-based approach to constructing classification and identification models for compound objects. By an object we mean an element of real world, which can be stored as a data object represented using ontology specified in alignment with domain knowledge about a given problem [2]. By a compound object we mean an object, which combines a plurality of objects within ontology-definable structure. Such structure can be further used to synthesize similarities basing on analysis of object components [3].

The proposed framework is based on hierarchical comparisons of investigated objects with reference sets reflecting different levels of object structures. As a case study, we consider a task of identifying components in texts representing bibliography items [4]. The process of assigning dynamically derived text

This work was partly supported by Polish National Centre for Research and Development (NCBiR) grants O ROB/0010/03/001 in frame of Defence and Security Programmes and Projects and PBS2/B9/20/2013 in frame of Applied Research Programmes, as well as by Polish National Science Centre (NCN) grants DEC-2012/05/B/ST6/03215 and DEC-2013/09/B/ST6/01568.

M. Kryszkiewicz et al. (Eds.): PReMI 2015, LNCS 9124, pp. 345–353, 2015.
DOI: 10.1007/978-3-319-19941-2_33

fragments to particular categories relies on comparing them with a reference database of publications treated as compound objects [5]. It can be envisioned as a resemblance-based recognition method, where similarity to labeled objects enables us to extrapolate assignments onto new items.

In our previous research, we investigated a number of applications of compound object comparators going beyond typical classification tasks [6]. We also attempted to provide possibly complete description of how to construct networks of comparators in practice. However, a deeper analysis of theoretical foundations of our approach has been still missing. Thus, the main focus in this paper is on mathematical interpretation of transmitting comparisons through a network, using mainly terminology of fuzzy sets and relations [7].

The paper is organized as follows: Sect. 2 introduces basic notions corresponding to a single comparator of compound objects. Section 3 establishes foundations for similarity-related operations, which occur inside multilayered networks of comparators. Section 4 recalls and clarifies already-mentioned experiments with analysis of bibliography items. Section 5 summarizes our work and specifies some research directions for nearest future.

One can think about our approach as analogous to feedforward neural networks [8]. However, comparators work with two kinds of information: about an input object described by its possible structural characteristics and attribute values, and about its similarities to reference objects produced as outputs of previous network layers. Coexistence of these two components makes our model unique. On the other hand, it is certainly useful to compare it with other frameworks, such as those developed using already-mentioned fuzzy sets [9], those based on rough sets and rough mereology [10] and others.

2 Basics of Comparators

Comparator *com* computes a vector of similarities of an input object $u \in U$ to elements of a subset $X(u)$ of reference set ref. The aim of *com* is to narrow down the space of reference objects comparable to a given u. Such output can be used as final result of a comparison module embedded into a bigger application, it can be also combined with outputs of other comparators or transmitted to comparators in the next layer of a more compound network.

Comparator *com* receives a value of u on an attribute a, denoted as $a(u)$, and compares it to values $a(x)$ for reference objects $x \in X(u)$. The choice of a and other parameters inside *com* are based on domain knowledge about a given problem. A content of $X(u) \subseteq ref$ may depend on outcomes of other comparators in a network. At the start of computations, we assume $X(u) = ref$. Given input set U, we will represent *com* as function $\mu_{com} : U \times 2^{ref} \rightarrow [0,1]^{ref}$, where $[0,1]^{ref}$ denotes all fuzzy sets over discrete domain ref.

Fuzzy sets $\mu_{com}(u, X(u))$ are computed in several steps. First, $u \in U$ is compared to each $x \in X(u)$ separately. The result of comparison can be represented as fuzzy relation $\mu_a(u, x)$ between values (representations) of a for u and x. Quantities of $\mu_a(u, x)$ are then filtered in two stages. First, we check through

predefined exception rules to exclude reference objects, which should not be compared with u based on available information. Secondly, we check whether the remaining quantities are not lower than an activation threshold $p > 0$. Overall, we can use a formula modifying initial μ_a as follows:

$$\mu_a^*(u,x) = \begin{cases} \mu_a(u,x) \text{ if } x \in X(u) \text{ and } \mu_a(u,x) \geq p \\ \qquad\quad \text{and there are no rules which} \\ \qquad\quad \text{disallow comparing } u \text{ with } x \\ 0 \qquad\quad \text{otherwise} \end{cases} \tag{1}$$

The next two steps, represented as filtering function $f : [0,1]^{ref} \to [0,1]^{ref}$ and sharpening function $s : [0,1]^{ref} \to [0,1]^{ref}$, aim at further filtration of similarity coefficients by their mutual comparisons and strengthening the highest remaining weights. For this purpose, it is more convenient to think about vector $\mu_a^*(u)$ with coordinates defined as $\mu_a^*(u)[i] = \mu_a^*(u, x_i)$, $i = 1, ..., |ref|$.

The role of f is to increase the amount of zero coordinates of $\mu_a^*(u)$. For example, one can set to 0 all elements, which are not among n highest similarity scores, for $n \geq 1$. Calculation of most of filtering functions considered in our previous research can be optimized by splitting coordinates onto blocks, deriving f concurrently and merging results. It is particularly important for applications, which require operating with large cardinalities of ref.

The role of s is to introduce non-linearity, whose benefits can be compared to a usage of exponential functions in feedforward neural networks. Let us put $\mu_a^f(u) = f(\mu_a^*(u))$. The following formula for s works only with non-zero coefficients and keeps maximal values of $\mu_a^f(u)$ unchanged. These properties are important for both the speed and accuracy of computing.

$$s(\mu_a^f(u))[i] = \begin{cases} \max_u \cdot e^{\mu_a^f(u)[i] - \max_u} \text{ if } \mu_a^f(u)[i] > 0 \\ 0 \qquad\qquad\qquad\quad \text{otherwise} \end{cases} \tag{2}$$

where $\max_u = \max_i \mu_a^f(u)[i]$. Derivation of $s(f(\mu_a^*(u)))$ can be also expressed using operations on fuzzy sets and relations, where similarities between u and reference objects correspond to fuzzy membership degrees.

Vector $s(f(\mu_a^*(u)))$ can be treated as output of single comparator. In a larger network, there can exist several interrelated comparators looking at the same ref by means of different attributes. Let us denote by $com_1, ..., com_l$ a set of such comparators, working with $a_1, ..., a_l$ respectively. We put

$$\mu_{com}(u, X(u)) = \overline{\left(s(f(\mu_{a_1}^*(u))), ..., s(f(\mu_{a_l}^*(u)))\right)} \tag{3}$$

as output of composite comparator com containing $com_1, ..., com_l$ as its parts. The role of function $\overline{(...)}$ is to synthesize local outcomes in order to send further a unique signal related to ref. Such synthesis can be based on fuzzy t-norms and s-norms, statistical tools, election algorithms and so on [11].

3 Comparator Networks

Let us denote a network of comparators by net. Performance of net can be characterized analogously to a single comparator, by a function $\mu_{net} : U \to$

$[0,1]^{ref}$. Overall outcome can be utilized directly in object identification process or, e.g., as an input to a similarity-based classifier, which checks sums of weights of reference objects dropping into particular decision classes.

Structure of *net* is similar to multilayered feedforward neural networks, although transmitted signals and calculations inside nodes are different. Each layer of *net* contains a set of comparators and a specific translating/aggregating mechanism. Comparators run in parallel, usually basing on different attributes. Thus, from computational perspective, we can see that concurrency can be achieved both at the level of single comparators and their larger groups. The role of translator is to convert comparator outputs to information about reference objects that would be useful for the next layer. The role of aggregator is to choose the most likely outputs of the translator, in case there was any non-uniqueness in assigning information about input objects to comparators. We will see that those roles can be interpreted using fuzzy t-norms and fuzzy s-norms.

Each object is described using ontologies defined by concepts and relationships between them. Given a hierarchy of concepts, one can consider relationships of generalization and decomposition. Generalization is a relationship of being a sub-object of another object, while decomposition is a relationship of being a parent (super-object) of a set of sub-objects. Particular layers of *net* usually correspond to hierarchy levels, so transitions between them correspond to generalization of decomposition. This affects the way of handling both input and reference objects, as well as modeling similarities between them.

Consequently, in a single *net*, different comparators can refer to different types and levels of reference (sub-)objects, using different attributes and parameters. Thus, the first task is to extract for a given u its structural representation, i.e., all its parts and their corresponding attribute values. Moreover, it is not always obvious which parts of u should be compared to particular reference sets. In such cases, a single u can yield multiple possible combinations of assignments of its parts to particular comparators. All such alternative representations, denoted as u', should be processed through the first layers and, later, the most probable assignments of u's parts to particular categories of reference objects can be derived. One can think about collections of possible representations u' as information granules $g(u)$ created around input objects $u \in U$ [12].

Inputs to each layer are determined by values of attributes for $u \in U$ or its sub-objects. However, subsets of reference (sub-)objects, which u is going to be compared to, are induced dynamically by comparators in previous layers. In the simplest scenario, comparators in preliminary layers aim at reducing subsets of potentially comparable reference objects using relatively easily-computable attributes, leaving more complex calculations to further layers, where the number of reference items to be compared is already decreased. In other cases, initial layers work with attributes specified for sub-objects, producing vectors of similarities that need translation to the level of similarities between more compound objects, whose attributes are analyzed later. However, the complexity does not need to grow with consecutive layers. In some applications, the first layers can work with relatively basic attributes of compound objects, whose similarities are then translated to lower structural levels for detailed processing.

Types of reference objects can vary from layer to layer or even within a single layer. Comparators in a given layer usually refer to entities at the same level of ontology-based hierarchy of considered objects. However, a given hierarchy level can include multiple types of entities. Let us denote by $\mu^k_{net}(u)$ an outcome of the k-th layer for input object $u \in U$, after applying above-mentioned operations of translation and aggregation. Denote by $ref^{k+1}_1, ..., ref^{k+1}_{m(k+1)}$ reference sets used by comparators in the $(k+1)$-th layer. Our goal in this section is to specify function $\mu^k_{net} : U \to [0,1]^{ref^{k+1}_1} \times ... \times [0,1]^{ref^{k+1}_{m(k+1)}}$, which takes into account similarity vectors obtained from comparators in the k-th layer. Once we have $\mu^k_{net}(u)$, we can forward it as a signal granule and prepare subsets $X(u)^{k+1}_1 \subseteq ref^{k+1}_1, ..., X(u)^{k+1}_{m(k+1)} \subseteq ref^{k+1}_{m(k+1)}$ to be utilized by next comparators. Those two types of granules – the above signal granule and previously-mentioned information granule $g(u)$ – illustrate a twofold way of operating with information about objects throughout networks of comparators.

The central part of μ^k_{net} is matrix M^k_{net} with dimensions $|ref^k_1|+...+|ref^k_{m(k)}|$ and $|ref^{k+1}_1| + ... + |ref^{k+1}_{m(k+1)}|$, which links the k-th and the $(k+1)$-th layers of *net*. In its simplest implementation, it is a sparse boolean matrix encoding these of combinations of reference (sub-)objects in sets $ref^k_1, ..., ref^k_{m(k)}$ and $ref^{k+1}_1, ..., ref^{k+1}_{m(k+1)}$, which structurally correspond to each other. Matrices are created during the process of defining reference sets, whose elements are decomposed due to their ontology-based specifications. Connections can be also additionally weighted with degrees expressing, e.g., to what extent particular sub-objects should influence similarities between their parents.

Translation can be executed as a product of M^k_{net} with concatenated vectors of similarities obtained as outputs of comparators $com^k_1, ..., com^k_{m(k)}$ in the k-th layer, for each of possible representations of u gathered in information granule $g(u)$. Let us enumerate all such representations as $u'_1, ..., u'_{|g(u)|}$ and denote by $G^k_{net}(u)$ the matrix of all possible output combinations, that is:

$$G^k_{net}(u) = \begin{bmatrix} \mu_{com^k_1}(u'_1)[1] & \dots & \mu_{com^k_1}(u'_{|g(u)|})[1] \\ \vdots & \ddots & \vdots \\ \mu_{com^k_1}(u'_1)\left[|ref^k_1|\right] & \dots & \mu_{com^k_1}(u'_{|g(u)|})\left[|ref^k_1|\right] \\ \mu_{com^k_2}(u'_1)[1] & \dots & \mu_{com^k_2}(u'_{|g(u)|})[1] \\ \vdots & \ddots & \vdots \\ \mu_{com^k_{m(k)}}(u'_1)\left[|ref^k_{m(k)}|\right] & \dots & \mu_{com^k_{m(k)}}(u'_{|g(u)|})\left[|ref^k_{m(k)}|\right] \end{bmatrix} \quad (4)$$

We can represent the mechanism for computing $\mu^k_{net}(u)$ as follows:

$$\mu^k_{net}(u)[i] = \max_j \min\left((M^k_{net}G^k_{net}(u))[i][j], 1\right) \quad (5)$$

where $[i][j]$ denotes coordinates of matrix $M^k_{net}G^k_{net}(u)$. Surely, specification of required operations in terms of matrices and vectors helps in efficient implementation. On the other hand, we can see below that these calculations can be indeed interpreted by means of well-known t-norms and s-norms.

Firstly, for a given $u'_j \in g(u)$, column $(M^k_{net} G^k_{net}(u))[j]$ represents possible similarities of u to reference objects in the $(k+1)$-th layer. Each of those similarities is computed as a sum of similarities between components of u (distributed among comparators according to combination u'_j) and reference objects in the k-th layer. If it exceeds 1, then of course we cut it down. Thus, similarities between objects at the $(k+1)$-th layer are computed as Łukasiewicz's t-norm of similarities between the corresponding objects at the k-th layer.

Secondly, in order to finally assess similarity of u to a given reference object in the $(k+1)$-th layer, we look at all combinations in $g(u)$ and choose the maximum possible score. Thus, we follow Zadeh's s-norm. Intuitively, our usage of t-norm corresponds to taking a conjunction of component similarities in order to judge similarity between compound objects, while our usage of s-norm reflects a disjunction of all alternative ways of obtaining that similarity. From this perspective, our current implementation reflects one of possible specifications and other settings of t-norm and s-norm could be considered as well.

Surely, the above layout is still a kind of simplification. As noted in Sect. 2, some comparators can comprise of multiple sub-units referring to different attributes or even different types of objects. However, function-based interpretation of network performance enables to look at such composite cases as a recursive specification of how information is flowing. Moreover, it lets better understand how to adapt existing data-based learning approaches, such as error backpropagation in neural networks transmitting compound signals [13], which might be utilized, e.g., to adjust weights in translation matrices.

Actually, the topic of learning comparator networks is far wider. For example, parameters responsible for synthesis of partial outcomes of composite comparators, usage of layer outcomes to specify reference subsets for next layers, as well as aggregation of final network results can be all tuned by basing on, e.g., evolutionary algorithms [14]. Moreover, some attribute and object selection methods developed within already-mentioned framework of rough sets could be utilized to optimize configuration of comparators and reference sets [15].

4 Illustrative Example

Methods outlined in previous sections have been used in a number of academic and commercial projects. As a case study, let us discuss the task of analysis of bibliography items, described in more detail in [5]. The goal here is to determine structural patterns of references represented as unstructured texts, so their fragments get identified as members of classes such as author names, paper titles, publication dates and so on. The comparator-network-based solution aimed at this kind of text processing was designed as a component of the system responsible for indexing articles stored in scientific repositories [4].

As an example, the text "*Sosnowski, Ł.: Framework of Compound Object Comparators. Intelligent Decision Technologies (2015)*" should be recognized as aligned with structural pattern ATJY, where A, T, J and Y stand for authors, title, journal and year, respectively. Also, "*Sosnowski, Ł.*" should be identified as existing or added as new element of reference set of authors etc.

Such recognition process can be divided into preprocessing, parsing and classification. The first stage is responsible for filtering out completely useless characters (e.g.: exclamation marks). The second stage splits text onto potentially meaningful parts, basing on appropriate interpretation of punctuation and additional rules aiming at merging some of produced parts together and final cleaning. As a result, we obtain components for further usage.

For the third stage, we employ the network with input layer containing comparators corresponding to the following categories of reference objects acquired from the considered repository [4]: Authors (A), Book (B), Country (C), Doi (D), Journal (J), Pages (P), Proceedings (R), Series (S), Title (T), Volume (V), Year (Y). Different comparators work with different attributes. We assume that elements of reference sets are already correctly classified.

Comparator dedicated to authors includes sub-comparators looking at sorted initials (si), longest lengths of text fragments (ll) and full strings representing authors (au). Their similarity measures are as follows:

$$\begin{aligned} \mu_{si}(u,x) &= 1 - \frac{d_L(u,x)}{max\{n(u),n(x)\}} \\ \mu_{ll}(u,x) &= 1 - \frac{|n(u)-n(x)|}{max\{n(u),n(x)\}} \\ \mu_{au}(u,x) &= \frac{1+pos(u,x)-neg(u,x)}{2+pen(u,x)} \end{aligned} \tag{6}$$

where $n(x)$ denotes the length of x (if x and u are empty, then we put $\mu_{si}(u,x) = \mu_{ll}(u,x) = 1$), $d_L(u,x)$ denotes Levenshtein's edit distance, $pos(u,x)$ is the average similarity between tokens occurring within u and their corresponding best-matching tokens within x, $neg(u,x)$ is the ratio of tokens within u, for which we could not find any sufficiently similar tokens within x (please note that both pos and neg cannot exceed 1), and $pen(u,x)$ is the number of tokens within x, which were not chosen as best-matching counterparts for any tokens within u. Similarities used in other comparators are defined analogously, sometimes also involving comparisons of regular expression patterns.

For experiments, for training and testing, we use data sets with 132 and 268 texts, respectively. Training data set is used to fill in reference set of structural patterns. For each of 132 texts, we manually detect and classify their parts to A/B/C/D/J/P/R/S/T/V/Y categories and treat obtained sequences of codes (such as ATJY above) as structural reference objects.

Network is initiated with default activation thresholds $p = 0.5$ and uniform aggregation/translation weights. For each comparator and each text used for training, there is a dedicated unit test, which checks whether comparator's output includes correct answer. If not, then – depending on specific situation – reference set is enriched with a new object, which covers a given case, or comparator's activation threshold is set to be less rigorous.

Each of parsed test texts is processed in two stages. Firstly, our network completes part classification and produces the sets of candidate structural patterns. Then, the network conducts structure classification based on comparing those

candidates with patterns in structural reference set. Final result is an ordered subset of reference structural patterns.

Table 1. Upper left/right: best/worst results obtained when using part classification. Lower left/right: best/worst results obtained using complete process (part classification + structure classification). P_* (where $*$ is p or m), R_* and F_* stand for precision, recall and F_1-score, respectively. $_p$ and $_m$ stand for measurements related to outcomes of part classification and structure classification, respectively.

Pattern	P_p	R_p	F_p	P_m	R_m	F_m
ATR	1.00	1.00	1.00	0.75	1.00	0.86
RY	1.00	1.00	1.00	1.00	1.00	1.00
ATJVY	1.00	1.00	1.00	1.00	0.80	0.89
AT	1.00	0.98	0.99	0.61	0.92	0.73
ATRYP	1.00	0.93	0.96	1.00	0.73	0.83
ATVPYD	0.91	0.98	0.95	0.92	0.84	0.86
ATJVPYD	1.00	0.90	0.94	0.98	0.71	0.81
ATJVPY	1.00	0.86	0.92	1.00	0.58	0.72
AJVPYD	0.86	1.00	0.92	0.86	1.00	0.92
ATVPY	1.00	0.85	0.91	1.00	0.60	0.75

Pattern	P_p	R_p	F_p	P_m	R_m	F_m
ATC	0.33	0.67	0.44	0.00	0.00	0.00
ATYATYP	0.60	0.43	0.50	1.00	0.43	0.60
AYTRY	0.43	0.60	0.50	1.00	0.60	0.75
ATPYC	0.54	0.80	0.64	0.60	0.60	0.60
ATJVYP	1.00	0.50	0.65	0.50	0.25	0.33
ATVY	0.60	0.75	0.67	1.00	0.75	0.86
ATVYPR	0.56	0.83	0.67	1.00	0.50	0.67
ATJPY	0.94	0.58	0.70	0.89	0.60	0.71
ATRPYC	0.71	0.77	0.73	1.00	0.77	0.86
ATAT	0.57	1.00	0.73	0.00	0.00	0.00

Pattern	P_p	R_p	F_p	P_m	R_m	F_m
RY	1.00	1.00	1.00	1.00	1.00	1.00
ATRY	0.89	0.88	0.86	1.00	0.88	0.93
ATRPY	0.93	0.90	0.90	0.99	0.88	0.92
AJVPYD	0.86	1.00	0.92	0.86	1.00	0.92
ATRJPY	0.83	0.83	0.83	1.00	0.83	0.91
ATRPYCD	0.84	0.85	0.84	0.97	0.85	0.91
ATPYD	0.83	1.00	0.91	0.83	1.00	0.91
ATY	0.88	0.91	0.87	0.90	0.90	0.90
ATJVY	1.00	1.00	1.00	1.00	0.80	0.89
ATRP	1.00	0.75	0.86	0.88	0.88	0.88

Pattern	P_p	R_p	F_p	P_m	R_m	F_m
AYTJP	1.00	0.70	0.82	0.00	0.00	0.00
ATC	0.33	0.67	0.44	0.00	0.00	0.00
ATAT	0.57	1.00	0.73	0.00	0.00	0.00
ATJYR	1.00	0.60	0.75	0.00	0.00	0.00
AYT	0.94	0.87	0.87	0.13	0.13	0.13
ATJVYP	1.00	0.50	0.65	0.50	0.25	0.33
AYTP	0.73	0.85	0.76	0.37	0.30	0.33
ATYR	1.00	0.75	0.86	0.50	0.50	0.50
ATYP	0.83	0.69	0.75	0.61	0.47	0.53
ATPYC	0.54	0.80	0.64	0.60	0.60	0.60

Table 1 includes results in terms of standard evaluation measures, such as precision, recall and F_1-score [16]. It shows the best and the worst results for part classification (the first stage only) and complete solution (both stages mentioned above). Global average values of F_1-score are equal to 0.86 and 0.78 for the first case and the second case, respectively.

The reason for lower F_1-score in the second case is that some structural patterns obtained for test texts may not be present in structural reference sets, so performing structure classification is actually a harder task. It can also happen that inputs are corrupted or wrongly created, which is a bigger problem for entire texts than for their parts. Still, obtained results make it possible to use this solution in practice, if applied together with incremental methods for cleaning, unifying and extending reference sets.

5 Conclusion

Networks of comparators are useful for solving decision problems requiring similarity modelling. They are characterized by a common modular approach to various tasks, such as classification, identification, etc., based on comparator units and their corresponding reference sets. In this paper, we showed to what extent networks of comparators can be described using fuzzy set terminology and operations. We hope that reported mathematical foundations will lead toward new areas of applications of our methodology.

References

1. Tversky, A., Shafir, E.: Preference, Belief, and Similarity - Selected Writings. Bradford Books. MIT Press, Cambridge (2004)
2. Staab, S., Maedche, A.: Knowledge portals - ontologies at work. AI Mag. **22**(2), 63–75 (2001)
3. Schickel-Zuber, V., Faltings, B.: OSS - a semantic similarity function based on hierarchical ontologies. In: Proceedings of IJCAI, pp. 551–556 (2007)
4. Ślęzak, D., Stencel, K., Nguyen, H.S.: (No)SQL platform for scalable semantic processing of fast growing document repositories. ERCIM News **2012**(90), 50–51 (2012)
5. Sosnowski, Ł.: Framework of Compound Object Comparators. Intelligent Decision Technologies (2015). doi:10.3233/IDT-140229
6. Szczuka, M.S., Sosnowski, Ł., Krasuski, A., Kreński, K.: Using domain knowledge in initial stages of KDD - optimization of compound object processing. Fundam. Inform. **129**(4), 341–364 (2014)
7. Kacprzyk, J.: Multistage Fuzzy Control- A Model-based Approach to Fuzzy Control and Decision Making. Wiley, Chichester (2012)
8. Rutkowski, L.: Computational Intelligence - Methods and Techniques. Springer, Heidelberg (2008)
9. Marín, N., Medina, J.M., Pons, O., Sánchez, D., Vila, M.A.: Complex object comparison in a fuzzy context. Inf. Softw. Technol. **45**, 431–444 (2003)
10. Pawlak, Z., Skowron, A.: Rough sets - some extensions. Inf. Sci. **177**(1), 28–40 (2007)
11. Sosnowski, Ł., Pietruszka, A., Krasuski, A., Janusz, A.: A resemblance based approach for recognition of risks at a fire ground. In: Ślęzak, D., Schaefer, G., Vuong, S.T., Kim, Y.-S. (eds.) AMT 2014. LNCS, vol. 8610, pp. 559–570. Springer, Heidelberg (2014)
12. Zadeh, L.A.: Computing with Words - Principal Concepts and Ideas. Volume 277 of Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2012)
13. Szczuka, M.S., Ślęzak, D.: Feedforward neural networks for compound signals. Theoret. Comput. Sci. **412**(42), 5960–5973 (2011)
14. Stahl, A., Gabel, T.: Using evolution programs to learn local similarity measures. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689. Springer, Heidelberg (2003)
15. Janusz, A., Ślęzak, D., Nguyen, H.S.: Unsupervised similarity learning from textual data. Fundam. Inf. **119**(3–4), 319–336 (2012)
16. Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In: Procedings of CoNLL, pp. 155–158 (2002)