

A Less Cumulative Algorithm of Mining Linguistic Browsing Patterns in the World Wide Web

Krzysztof Dyczkowski

Adam Mickiewicz University
Faculty of Mathematics and Computer Science
Umultowska 87
Poznan, Poland
e-mail chris@amu.edu.pl

Abstract

Finding sequential patterns is one of important issues in data mining. This paper deals with linguistic (fuzzy) sequential patterns. The existing algorithms for discovering such patterns do involve usual sigma counts of fuzzy sets as measure of support. Unfortunately, a well-known side effect is then an undesirable cumulation of small membership values. We like to propose an improved approach based on generalized sigma counts, which in particular leads to a consequently lower cumulation, and considerably to better results. We test the modified algorithm using different generalized sigma counts based on different cardinality pattern functions.

Keywords: Data mining, Linguistic sequential patterns, Scalar Cardinality, Generalized sigma counts, Pattern function

1 Introduction

Internet servers especially WWW servers, generate a huge number of data. In particular, they log large amounts of information about connections, security events and users transfers. Finding regularities in such (usually) large data files leads to security improvement, brings knowledge about users behavior and, therefore, is of interest to administrators, marketing departments, etc.

There are two main approaches to discovering those singularities: association rules and sequential patterns (see [1, 2, 3, 4, 8, 9]). We like to focus on sequential patterns which are expression of the form $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_n$, where X_i is a set of some items. In the case of linguistic sequential patterns, the X_i 's are equipped with some linguistic labels. In particular, a linguistic browsing pattern in the WWW is a sequence of web pages to which linguistic terms expressing the

duration of browsing time are assigned. For instance, the linguistic browsing pattern $file1.html/short \rightarrow file2.html/long \rightarrow file3.html/medium$ says that a frequent behavior of users is that they watch the page $file1.html$ for a short time, then they watch $file2.html$ for a long time and, further, they watch $file3.html$ for a medium time. It was Hong et al. [7] (cf.[8]) who proposed an algorithm for finding such patterns which is an adaptation of the ideas by Agrawal and Shirkant [1, 2, 3]. The following are three main steps of that algorithm:

1. Create fuzzy sets from transactions data using given membership functions.
2. Generate large itemsets by calculating the fuzzy cardinality of each candidate itemset.
3. Generate iterative sequential patterns from the large itemsets found in step 2.

By the way Hong et al. proposed in [9] a GA algorithm which dynamically constructs membership functions to model linguistic terms of browsing time.

The Hong's algorithm from [7] uses the usual sigma count as a measure of support. Unfortunately, a well-known side effect is then an undesirable cumulation of small membership values which can lead to wrong conclusions. For example, a fuzzy set A with 100 elements with membership values equal to 0.01 has the sigma count equal to 1, which is not coherent with intuition (see [5]).

In this paper we like to propose an improved approach based on generalized sigma counts, which in particular leads to considerably lower cumulation and, consequently, to better results.

2 Generalized Sigma Counts

Our further discussion requires some notions related to cardinalities of fuzzy sets. FFS will denote the family

of all finite sets in a universe \mathcal{M} . Let us recall the axiomatic definition of scalar cardinality according to Wygralak [13, 14]

A function $\sigma : FFS \rightarrow [0, \infty)$ will be called a scalar cardinality if the following postulates are satisfied for each $a, b \in [0, 1]$, $A, B \in FFS$ and $x, y \in \mathcal{M}$:

- (SC1) $\sigma(1/x) = 1$,
- (SC2) $a \leq b \Rightarrow \sigma(a/x) \leq \sigma(b/y)$,
- (SC3) $A \cap B = 1_\emptyset \Rightarrow \sigma(A \cup B) = \sigma(A) + \sigma(B)$.

If σ does fulfill the above axioms, the number $\sigma(A)$ is said to be a scalar cardinality of A .

The following characterization correspond to scalar cardinalities defined via (SC1)-(SC3).

Theorem 1 *A mapping $\sigma : FFS \rightarrow [0, \infty)$ is a scalar cardinality iff*

$$\forall A \in FFS : \sigma(A) = \sum_{x \in \text{supp}(A)} f(A(x)), \quad (1)$$

where $f : [0, 1] \rightarrow [0, 1]$ is nondecreasing and $f(0) = 0$, $f(1) = 1$.

Each function f from Theorem 1 is called a cardinality pattern or pattern function. Since $f = id$ gives as $\sigma(A)$ the usual sigma count of A from [16], each $\sigma(A)$ from (1) will be called a generalized sigma count of A . To emphasize which pattern function f is used to generate a generalized sigma count, we will write $SC_f(A)$ instead of $\sigma(A)$.

In the following, we present some basic examples of pattern function together with the resulting generalized sigma counts. They are instances of pattern functions leading to a reduction of accumulation of small membership degrees.

Example 2.1 *Let $t \in (0, 1]$*

$$f_1(a) = \begin{cases} 1, & \text{if } a \geq t, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

for $a \in [0, 1]$. One can also see that choice of even door-sill 1 is justified.

It is easy to see that $SC_{f_1}(A) = |A_t|$, where A_t denotes the t -cut set of A . f_1 reduces the cumulation effect by reducing to zero each membership value which is less than a fixed threshold t , and by increasing to 1 each membership value $\geq t$ (see Fig. 1).

Example 2.2 *Let*

$$f_2(a) = a^p \quad (3)$$

for $a \in [0, 1]$, $p > 0$. (see Fig. 2)

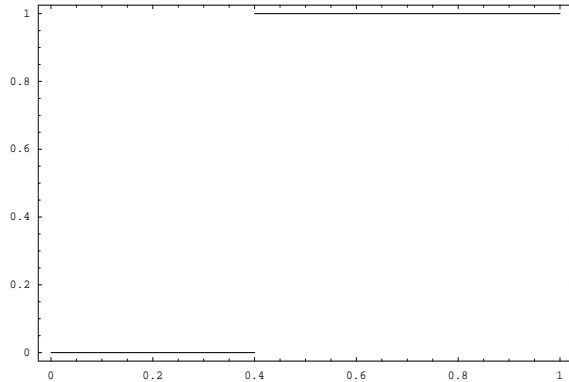


Figure 1: Pattern function f_1 with $t = 0.4$

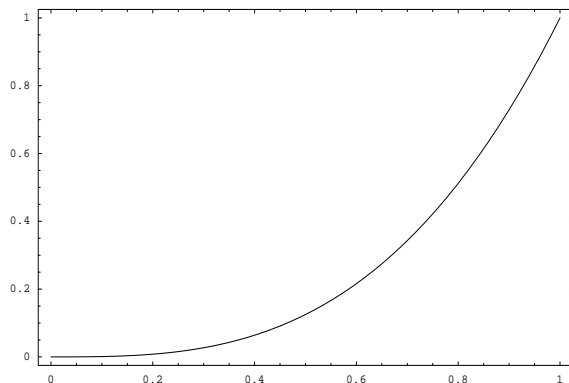


Figure 2: Pattern function f_2 with $p = 4$

Obviously, $p = 1$ gives the usual sigma count.

Example 2.3 *Let $b, c \in (0, 1)$*

$$f_3(a) = \begin{cases} 0, & \text{if } a \leq b, \\ a & \text{if } a \in (b, c), \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

for $a \in [0, 1]$.

In this case, we are using two threshold levels b and c . The threshold b determines which low degrees of the membership are supposed to be 0 and c determines which big values of the membership are supposed to be replaced by 1. The values of the membership between b and c stays untouched. (see Fig. 3).

Example 2.4 *Let $b, c, d \in (0, 1]$ and $b < c < d$*

$$f_4(a) = \begin{cases} 0, & \text{if } a \leq b, \\ 2\left(\frac{a-b}{c}\right)^2, & \text{if } a \in (b, c], \\ 1 - 2\left(\frac{a-d}{c}\right)^2, & \text{if } a \in (c, d), \\ 1, & \text{otherwise.} \end{cases} \quad (5)$$

Function f_3 (4) is discontinuous contrast enhancement function. f_4 (5) is more sophisticated continuous variant of such function (see Fig 4). This kind of pattern

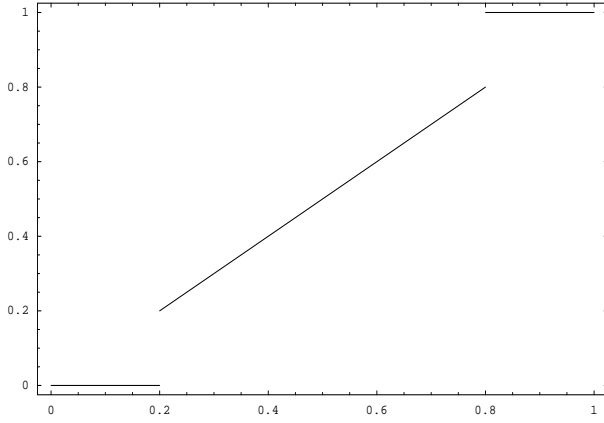


Figure 3: Pattern function f_3 for $b = 0.2, c = 0.8$

function promote higher values of membership function. We will take an advantage of this pattern function in examples.

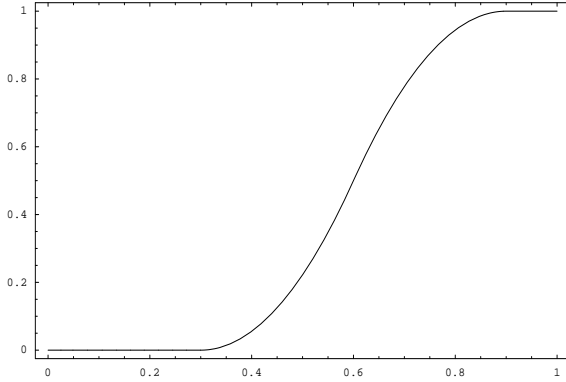


Figure 4: Pattern function f_4 with $b = 0.3, c = 0.6, d = 0.9$

Example 2.5 Let $b \in (0, 1]$

$$f_5(a) = \begin{cases} 0, & \text{if } a < b, \\ a, & \text{otherwise.} \end{cases} \quad (6)$$

for $a \in [0, 1]$.

In this case pattern function (6) works as identity function (as a usual sigma count) except values less than threshold point b which are treated as 0 (see Fig. 5). So b decides how small values will not be cumulated.

Example 2.6 Let $p \in (0, 1]$

$$f_6(a) = 1 - (1 - a)^p \quad (7)$$

for $a \in [0, 1]$.

Worth mentioning is that each normed generator of a nonstrict Archimedean t-conorm is a pattern function

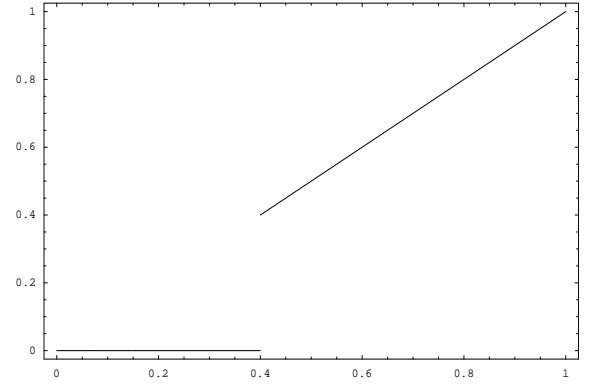


Figure 5: Pattern function f_5 with $b = 0.4$

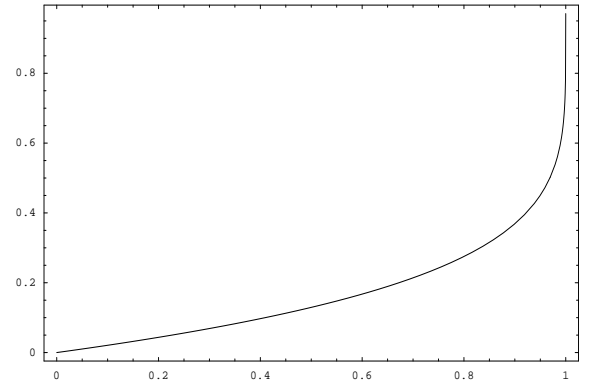


Figure 6: Pattern function f_6 with $a = 0.2$

(see [13]). As example, take the normed generator (7) of the Schweizer t-conorm $s_{S,\lambda}(a, b) := 1 - (0 \vee ((1 - a)^\lambda + (1 - b)^\lambda - 1))^{\frac{1}{\lambda}}$ whit $\lambda > 0$ (see Fig. 6).

Notice that function f_2 (3) is just normed generator of Yager t-conorm.

To show how values of cardinality depends on choice of cardinality function we show next example.

Example 2.7 Let us consider a fuzzy set A to which 50 elements belong to degree 0.1, 30 elements to degree 0.2, 10 elements to degree 0.3, 6 elements to degree 0.4, 2 elements to degree 0.8 and 2 elements to degree 0. The sigma count of this fuzzy set equals $SC_{id}(A) = 19.8$, which seems to be a inflated result in comparison with our intuition. Computing generalized sigma counts, we get more adequate results, for instance

$$\begin{aligned} SC_{f_1}(A) &= 10, \\ SC_{f_2}(A) &= 3.426, \\ SC_{f_3}(A) &= 9.4, \\ SC_{f_4}(A) &= 4.222, \\ SC_{f_5}(A) &= 5.8, \\ SC_{f_6}(A) &= 4.912, \end{aligned}$$

where the cumulation effect is reduced. As we see, the choice of an appropriate pattern function is a key issue when using generalized sigma counts.

3 An Improved algorithm of Mining Sequential Patterns

For the sake of simplicity, the algorithm will be presented in non-formalized form. As input data we take log file from WWW server. It is a set U of n transactions, where n can be large. Each transaction contains browsed information about opened files from WWW server. For the mining process the following files of transactions will be of interest: datetime, source IP and name of opened file. At the beginning we need to

Table 1: Example of transactions in the log

Client IP	Time	Web Page
194.114.147.2	22:03:57	index.html
194.114.147.2	22:04:11	offer.html
194.114.147.2	22:04:25	rocker.html
194.114.147.2	22:04:33	b8250.html
194.149.88.10	12:51:50	index.html
194.149.88.10	12:52:06	offer.html
194.149.88.10	12:52:28	rocker.html
194.149.88.10	12:52:28	bench.html
195.46.43.144	11:20:11	index.html
195.46.43.144	11:20:15	offer.html
195.46.43.144	11:20:20	rocker.html
212.122.214.145	20:07:18	index.html
212.122.214.145	20:07:23	offer.html
212.122.214.145	20:07:29	bench.html
212.122.214.145	20:07:40	rocker.html
212.122.214.145	20:07:48	chair.html
212.122.214.145	20:07:54	a0910.html

preprocess log data in three steps:

1. Select from log only transactions containing files that are web pages for example such as ".html", ".php", ".asp" etc. Example of part of such transformed log is shown in Table 1.
2. Replace each IP by unique integer (called encoded client ID). For convenience we will represent all files as capital letters eg. file "index.htm" as "A", "offer.html" as "B" and so on.
3. Sort by client ID and datetime fields.
4. For each client calculate the time durations of browsing each web page as time to open the next file. Last files for particular client ID will be omitted, because of the lack of information about the moment of disconnecting.

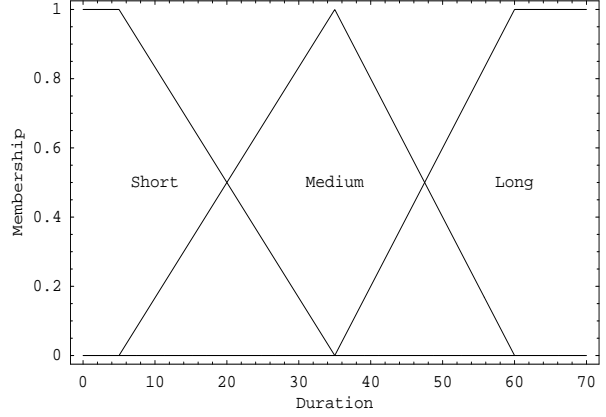


Figure 7: Example of linguistic terms of browsing time: Short, Medium, Long

Table 2: Example of transformed log into linguistic sequences

ID	Page	Short	Medium	Long
1	A	0,080	0,920	0,000
1	B	0,160	0,840	0,000
1	C	0,000	0,833	0,167
2	A	0,800	0,200	0,000
2	B	0,760	0,240	0,000
2	D	0,000	0,700	0,300
3	A	0,960	0,040	0,000
3	B	1,000	0,000	0,000
3	E	0,000	0,900	0,100
3	D	0,800	0,200	0,000
4	A	1,000	0,000	0,000
4	B	0,960	0,040	0,000
4	C	0,640	0,360	0,000
4	E	1,000	0,000	0,000
4	D	0,000	0,300	0,700
4	E	0,840	0,160	0,000
4	C	1,000	0,000	0,000
5	A	0,760	0,240	0,000
5	B	0,480	0,520	0,000
5	F	0,600	0,400	0,000
6	A	0,760	0,240	0,000
6	B	0,080	0,920	0,000
6	F	0,920	0,080	0,000
6	D	1,000	0,000	0,000
7	A	0,480	0,520	0,000
7	B	0,320	0,680	0,000
7	F	0,000	0,833	0,167
7	C	0,000	0,867	0,133
7	E	0,520	0,480	0,000

5. Construct browsing sequence by transforming each transaction into a fuzzy set $(\frac{x_1}{short} + \frac{x_2}{medium} + \frac{x_3}{long})$ using the given membership functions. It can be functions proposed in Figure 7.

Such linguistic browsing patterns will be an input for the mining algorithm described below:

1. Calculate membership value of each region for each browsing sequence using a fuzzy sum operator. A region is a web page with a linguistic value, for example: A.Short, B.Long, etc. For simplicity, we apply the max operation which, however, could be replaced by any t-conorm.
2. Calculate scalar cardinality of each region in all browsing sequences. To avoid problems with cumulating small values we use a generalized sigma count with a given pattern function.

For each file, select region with the value of cardinality greater than a fixed threshold α . We will use $\alpha = 1$. In [7], this step relies on choosing regions with the sigma count greater than or equal to α , where $\alpha = 2$.

We collect all selected regions in the set L_1 which is called the large itemset.

3. If L_1 is empty, we stop the algorithm. Otherwise, we put $k := 2$, which denotes the length of current sequence, and we do following.
4. Generate the set of candidates C_k consisting of pairs of all possible combinations of elements from L_{k-1} .
5. For each candidate from set C_k we calculate membership value for each user sequence using a t-norm as intersection operator. Let us use the minimum operation for simplicity.
6. Calculate a generalized scalar cardinality of each k-sequence in C_k in the way shown in Step 2 with given cardinality pattern. We choose only these sequences which have counts greater than given threshold point α . The resulting set is denoted by L_k .
7. If L_k is not empty we construct set C_{k+1} from L_k by combining all possible k-sequences with sequences from L_2 , and we go to Step 5 with $k:=k+1$.
8. Finally, take L_k as the set of longest sequences found by the algorithm.

4 Examples of Mining with Various Generalized Sigma Counts

To show the influence of the choice of a pattern function to the mining process, let us present some examples. We will use symbols "A.S", "A.M", "A.L" to denote linguistic items, where "A" represents file name

and "S" stands for "Short", "M" for "Medium", "L" for "Long". All computations refer to the small data set from Table 2. We put threshold point $\alpha = 1$ and use the t-norm minimum together with the t-conorm maximum.

Example 4.1 *The values given in brackets are values of generalized cardinality.*

1. Sets L_1 and L_2 generated by means of the standard sigma count SC_{id}

Set L_1
A.S (4.84), B.S (3.76), C.M (2.06), D.S (1.80), E.M (1.54), F.S (1.52)
Set L_2
A.S→B.S (3.640)

2. Sets L_1 , L_2 and L_3 generated using a SC_{f_1} with $b = 0.4$

Set L_1
A.S (6.00), B.M (5.00), C.M (2.00), D.S (2.00) E.S (2.00), F.S (2.00)
Set L_2
A.S→B.M (3.00), B.M→F.S (2.00)
Set L_3
A.S→B.M→F.S (2.00)

3. Sets L_1 and L_2 generated by means of SC_{f_2} with $p = 4$

Set L_1
A.S (2.98), B.S (2.25), C.M (1.06), D.S (1.41) E.S (1.07)
Set L_2
A.S→B.S (2.09)

4. Sets L_1 , L_2 and L_3 generated by means of SC_{f_3} with parameters as in Example 2.3

Set L_1
A.S (5.00), B.M (3.87), C.M (2.36), D.S (2.00) E.S (1.52), F.S (1.60)
Set L_2
A.S→B.M (2.00), B.M→F.S (1.52)
Set L_3
A.S→B.M→F.S (1.28)

5. Sets L_1 , L_2 and L_3 generated by means of SC_{f_4} with parameters as in Example 2.4

Set L_1
A.S (4.91), B.M (3.08), C.M (1.99), D.S (1.94) E.S (1.27), F.S (1.50)
Set L_2
A.S→B.M (1.34), B.M→F.S (1.27)
Set L_3
A.S→B.M→F.S (1.16)

Set L_{10}
S.S→S2.S→T.S→T2.S→T3.S→T4.S→ B.S→B2.S→B3.S→B4.S (1.12)

6. Set L_1 , L_2 and L_3 generated by means of SC_{f_5} with $b = 0.4$

Set L_1
A.S (4.76), B.M (3.39), C.M (1.70), D.S (1.80) E.S (1.52), F.S (1.52)
Set L_2
A.S→B.M (1.76), B.M→F.S (1.44)
Set L_3
A.S→B.M→F.S (1.28)

7. Set L_1 and L_2 generated by means of SC_{f_6} with $b = 0.2$

Set L_1
A.S (2.38), B.S (1.97), D.S (1.27) E.S (1.14)
Set L_2
A.S→B.S (1.43)

As we can see, the choice of the cardinality function plays a significant role. In the first example with the usual sigma count, cardinality values are relatively high but the length of maximal sequence is 2. In the next examples, values of cardinalities are smaller but sequences found are longer.

5 A Real Data Mining Example

Our mining algorithm, involving generalized sigma counts, and its implementation, have also been tested on real log data. Let us show some results for a log set of about 5.000 transactions concerning 900 clients who browsed about 200 web pages. We present only final sets of longest browsing patterns found for each cardinality pattern. As in previous example, we use the minimum and maximum operators, and $\alpha = 1$ as the threshold level.

Example 5.1 As previously, values in brackets are cardinalities.

1. The use of SC_{id} (the usual sigma count)

2. SC_{f_1} with $t = 0.4$

Set L_5
A.S→S.S→T.S→B.S→W.S (4.00)

3. SC_{f_2} with $p = 4$

Set L_5
S.S→E1.S→E2.S→T.S→B.S (1.06)
W.S→S.S→A.S→C.S→B.S (1.04)

4. SC_{f_3} with $b = 0.2$, $c = 0.8$

Set L_4
F3.S→C3.S→C2.S→C.S (1.12)
G.L→H.L→O.S→C.S (1.67)
C2.S→C3.S→L.S→C4.S→ (1.64)
B4.S→B5.S→F.S→V.S (1.20)
S.S→A.S→C.S→B.S (2.00)
S.S→T.S→B.S→W.S (8.00)

5. SC_{f_4} with $b = 0.3$, $c = 0.6$, $d = 0.9$

Set L_7
A.S→S.S→S2.S→T.S→T2.S→T3.S→T4.S (1.44)
S2.S→T.S→T2.S→T3.S→T4.S→B.S→B2.S (1.03)

6. SC_{f_5} with $b = 0.4$

Set L_5
O.S→B.S→T.S→S.S→A.S (1.44)

7. SC_{f_6} with $b = 0.2$

Set L_5
B.S→B2.S→B3.S→B4.S→B5.S (8.89)
T.S→T2.S→T3.S→T4.S→M.M (1.50)
S2.S→T.S→T2.S→T3.S→T4.S (1.05)

One sees that the usual sigma count leads to one long browsing pattern whose support is however weak, which is a consequence of the cumulation effect.

6 Conclusions

We presented a modified algorithm for mining linguistic browsing patterns in the WWW. Using generalized sigma counts we can reduce the effect of cumulating small values of membership function.

In our future work, we like to investigate methodological issues related to the choice of pattern functions generating generalized sigma counts. Also, we like to examine other t-norms and t-conorms for numerical interpretation of logical connectives in the mining procedure.

References

- [1] Agrawal R, Imielinski T, Swami A., Mining association rules between sets of items in large database, The 1993 ACM SIGMOD Conference, 207-216, 1993
- [2] Agrawal R, Srikant R., Fast algorithm for mining association rules, The International Conference on Very Large Data Bases, 487-499, 1994
- [3] Agrawal R, Srikant R., Mining sequential patterns, The Eleventh International Conference on Data Engineering, 314, 1995
- [4] Cooley R, Mobasher B., Srivastava J., Web mining: information and pattern discovery on the world wide web, Ninth IEEE International Conference on Tools with Artificial Intelligence, 558-567, 1997
- [5] Dubios D., Prade H., Sudkamp T., On the Representation, Measurement, and Discovery of Fuzzy Associations, IEEE Transactions on Fuzzy Systems, vol 13, 2:250-262, 2005
- [6] Fiot C., Laurent A., Teisseire M., Web Access Log Mining With Soft Sequential Patterns, 7th International FLINS Conference on Applied Artificial Intelligence (FLINS'06), 2006.
- [7] Hong T.P., Lin K.Y., Wang S.L., Mining linguistic browsing patterns in the world wide web, Soft Computing 6:329-336, Springer-Verlag, 2002
- [8] Hong T.P., Lin K.Y., Wang S.L., Mining fuzzy sequential patterns from quantitative transactions, Soft Computing 10:925-932, 2006
- [9] Hong T.P., Chen C.H., Wu Y.L., Lee Y.C., A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions, Soft Computing, 10:1091-1101, 2006
- [10] Larose D. T., Discovering Knowledge in Data: An Introduction to Data Mining, Wiley, 2005.
- [11] Klement E.P., Mesiar R., Pap E., Triangular Norms, Kluwer Academic Publishers, 2000.
- [12] Wygralak M., Triangular operations, negations, and scalar cardinality of fuzzy set, in: L.A. Zadeh, J. Kacprzyk (eds.), Computing with Words in Information/Intelligent Systems Vol. 1. Foundations, Physica-Verlag, Heidelberg New York, 326-341, 1999.
- [13] Wygralak M., An axiomatic approach to scalar cardinalities of fuzzy sets, Fuzzy Sets and Systems, 110, 175-179, 2000.
- [14] Wygralak M., Cardinalities of Fuzzy Sets Studies in Fuzziness and Soft Computing, vol 118, Springer Verlag, 2003.
- [15] Zadeh L. A., Fuzzy Sets, Inform. and Control, 8:338-353, 1965.
- [16] Zadeh L. A., From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions, IEEE Trans. on Circuits and Systems -I: Fundamental Theory and Appl., 45:105-119, 1999.