On the use of aggregation operators for location privacy

Aida Valls[1], Jordi Nin[2] and Vicenç Torra[2]

1. ITAKA Research Group - Intelligent Tech. for Advanced Knowledge Acquisition
Department of Computer Science and Mathematics
Universitat Rovira i Virgili
43007 Tarragona, Catalonia, Spain

2. IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB s/n
08193 Bellaterra, Catalonia, Spain

Email: aida.valls@urv.cat, {jnin,vtorra}@iiia.csic.es

Abstract—
Nowadays, the management of sequential and temporal data is an increasing need in many data mining processes. Therefore, the development of new privacy preserving data mining techniques for sequential data is a crucial need to ensure that sequence data analysis is performed without disclosure sensitive information. Although data analysis and protection are very different processes, they share a few common components such as similarity measurement.

In this paper we propose a new similarity function for categorical sequences of events based on OWA operators and fuzzy quantifiers. The main advantage of this new similarity function is the possibility of incorporating the user preferences in the similarity computation. We describe the implications of the application of different user preference policies in the similarity measurement when microaggregation, a well-known data protection method, is applied to sequential data.

Keywords— Microaggregation, Privacy, Sequence aggregation, OWA operators.

1 Introduction

The development of new methods for managing sequential and temporal relationships is becoming an strategic area. Nowadays, thanks to the new technologies, categorical sequences of data are easily available in many scenarios. Initially, sequential data was originated by the study of gene information (DNA), but novel classes of applications based on geo-spatial information are currently appearing [1]. For instance, many applications use personal or vehicular tracking data, this information allows us to find interesting patterns to be used in many different applications, such as traffic control, sustainable mobility management, accessibility of services or even tracking patients with Alzheimer [2]. This type of data is known as Event Sequences [3].

Classical data mining algorithms and decision making techniques were developed for static data (*i.e.* data whose feature values do not depend on time). Consequently, the sequential nature of temporal data makes difficult to apply classical data mining and decision making methods. Some areas of Artificial Intelligence have started to develop new methods to deal with event sequences, such as temporal data mining algorithms [4, 5] or sequence data analysis, both interested in studying how feature values change with regards to time, in order to identify interesting temporal patterns.

As in classical data mining, in order to ensure that sequence data analysis is performed without disclosure sensitive information of data owners, statistical disclosure control (SDC) [6] and privacy preserving data mining (PPDM) [7] techniques should be applied. In this way, the privacy of the individuals can be guaranteed. A very common way to achieve a certain level of privacy on a database is to ensure k-anonymity [8, 9]. Microaggregation [10, 11] is the standard SDC technique to achieve the k-anonymity in a database. Essentially, microaggregation builds clusters of at least k records and replaces the original records with the centroid of the cluster that the record belongs to. In this way, k-anonymity is ensured because in the protected database there are at least k identical records.

In this paper we address the problem of measuring the similarity between sequences of events. Similarity measurement is a key point in data analysis but also in microaggregation. This problem was already studied in [12], where a new approach to calculate the similarity for event sequences was presented: the *Ordered-based Sequence Similarity* (OSS). This similarity is greately improved in this paper. The new version presented in this paper, applies the OWA operator to let the user to define different preference policies for the similarity comparison of two event sequences, thus, it is called OSS_{OWA}. Preference policies can be expressed using different fuzzy measures. We will present the implications that some policies (fuzzy measure) have on the results of a microaggregation procedure with an illustrative example.

This paper is organized as follows. Firstly, in Section 2 we give some preliminaries about aggregation functions and microaggregation methods for SDC. Then, in Section 3 we present a detailed description of our new similarity function for categorical sequences of events. Section 4 provides some experiments combining our similarity function with microaggregation. Finally, Section 5 draws some conclusions and describes some lines for future work.

ISBN: 978-989-95079-6-8

2 Preliminaries

In this section we give a short overview of aggregation operators, focused on the OWA operator, and microaggregation (a data protection method).

2.1 Aggregation functions

Aggregation functions [13] are functions used for information fusion. They typically combine N data values supplied by N data sources into a single datum.

In this paper, we use them to define a new similarity measure among sequences of categorical events. In particular, we consider the Ordered Weighted Aggregation (OWA) operator.

Two different definitions for this operator can be found in the literature. One applicable when the number of data sources is known in advance, and another that does not require to know how many data sources are combined. We use this latter definition, which is based on fuzzy quantifiers.

Definition 1 *A function $Q : [0,1] \to [0,1]$ is a* regular monotonically non-decreasing fuzzy quantifier *(non-decreasing fuzzy quantifiers for short) if it satisfies: (i) $Q(0) = 0$; (ii) $Q(1) = 1$; (iii) $x > y$ implies $Q(x) \geq Q(y)$.*

Two examples of families of fuzzy quantifiers Q_1 and Q_2 are given below. Q_1 corresponds to Yager α-quantifiers, for $\alpha > 0$.

$$Q_1^{\alpha}(x) = x^{\alpha} \tag{1}$$

$$Q_2^{\alpha}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1/(1 + e^{(\alpha - x)*10}) \text{ for } \alpha > 0 & \text{if } 0 < x < 1 \\ 1 & \text{if } x = 1 \end{cases} \tag{2}$$

A graphical representation of these two fuzzy quantifiers is given in Figure 1 for some particular α. $\alpha = \{0.2, 0.4, \ldots, 1.8, 2.0\}$ are used for $Q_1^{\alpha}(x)$, and $\alpha = \{0, 0.1, \ldots 0.9\}$ are used for $Q_2^{\alpha}(x)$. We can observe that for small α values, the function increases quickly near $x = 0$, whereas the increase is smoothly for larger values of α.

Using fuzzy quantifiers, the OWA operator [14, 15] is defined as follows.

Definition 2 *Let Q be a non-decreasing fuzzy quantifier, then $OWA_Q : \mathbb{R}^N \to \mathbb{R}$ is an* Ordered Weighted Averaging (OWA) operator *if*

$$OWA_Q(a_1, ..., a_N) = \sum_{i=1}^{N} (Q(i/N) - Q((i-1)/N)) a_{\sigma(i)}$$

where σ is a permutation such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}$.

The interest of the OWA operators is that they permit the user to aggregate the values giving importance to large (or small) values. In the case of the quantifiers given above, the smaller the α is, the larger the importance for the largest values being aggregated. In contrast, the higher the α is, the lower the importance of the largest values (and the higher the importance given to low values). This different behaviour can be seen in Figure 1.

2.2 Microaggregation

Microaggregation [10, 11] is one of the most commonly employed data protection methods [16]. From the operational point of view, given a database with A attributes, microaggregation proceeds by building clusters of at least k records and, then, replacing each record by the centroid of the cluster to which the record belongs to. A certain level of privacy is ensured because for each attribute, there are always k records with an identical value. This assures the k-anonymity property at the attribute level [8, 9]. The goal of a microaggregation is to find clusters of at least k records while minimizing the total Sum of the Square Error:

$$SSE = \sum_{i=1}^{c} \sum_{x_{ij} \in C_i} (x_{ij} - \bar{x}_i)^T (x_{ij} - \bar{x}_i), \tag{3}$$

where c is the total number of clusters, C_i is the i-th cluster and $\bar{x}_i$ is the centroid of C_i. The restriction is $|C_i| \geq k$, for all $i = 1, \ldots, c$.

In the simplest case (when $A = 1$), there are polynomial time algorithms to obtain the optimal microaggregation [17]. However, for the general case ($A > 1$), the problem of finding the optimal microaggregation is NP-hard [18]. For this reason, microaggregation methods are heuristic.

It is worth to mention that most of the classical microaggregation techniques are based on the assumption that an average record for all the elements in the database can be computed, like MDAV algorithm [19]. For the case of dealing with sequences of events, this is not a feasible assumption, since sequences can be very diverse and the definition of a single average sequence is not meaningful. Other methods rely on an Euclidean space, which is also not the case of the event sequences space.

In [3] the use of hierarchical clustering algorithms for sequences of events was presented. Those methods are sequential algorithms based on a similarity measure between pairs of elements [20]. The result is a hierarchical tree of non-overlapping clusters that can be cut at any level to obtain a partition. For the experiments of this paper, the k-Sized Hierarchical Clustering method (KSHC) has been applied [21]. This method is a modification of the classical hierarchical sequential agglomerative non-overlapping clustering methods in data mining [20]. The clustering process is structured in two main steps: (1) the construction of a dissimilarity matrix for all pairs of individuals, using an appropriate similarity measure according to the type of data and problem characteristics and (2) the exploitation of the similarity matrix to build a tree of non-overlapping clusters. In this second stage, an iterative process is executed, which consists of finding the pair of elements (i.e. individuals or clusters) that are at minimum dissimilarity (i.e. the most similar) and build a new cluster with these elements. Then, those elements are removed from the dissimilarity matrix, and the dissimilarity of the new cluster with respect to the rest of elements is computed. Different methods for calculating this dissimilarity have been defined: Single Linkage (if the minimum distance with respect to the elements in the cluster is taken), Complete Linkage (if the maximum distance is taken), Average Linkage, Median Linkage, etc. The resulting tree can be cut at any level to produce a partition of the individuals. The KSHC approach keeps the

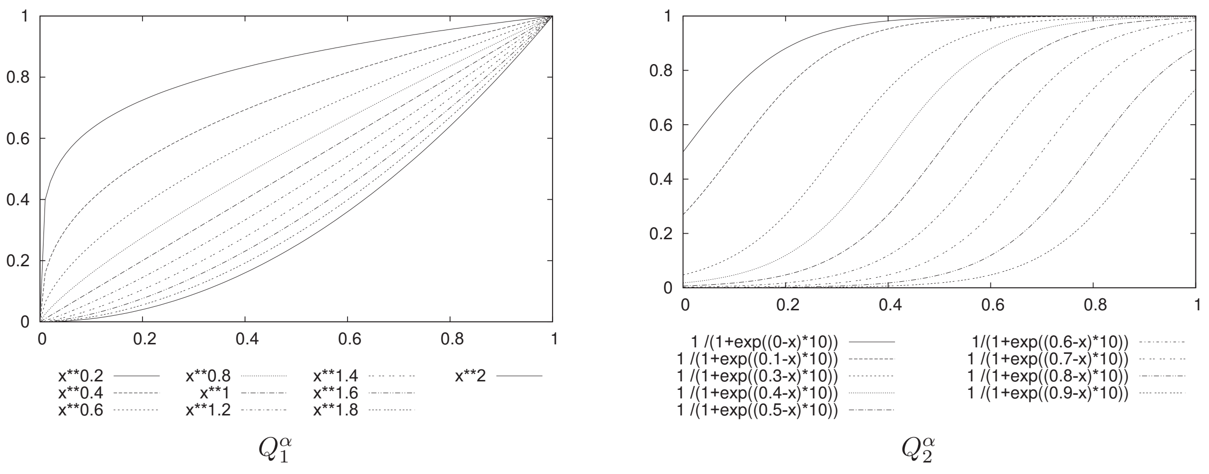

Figure 1: Graphical representation of Q_1^α and Q_2^α.

tradictional clustering process, but modifies it properly in order to produce a partition that respects the k-anonymity property.

2.3 The Cluster Prototype

Methods for defining cluster prototypes (or centroids) are needed in microaggregation as well as in some unsupervised machine learning techniques (*e.g.* in clustering). In general, a prototype is defined as a typical example, basis, or standard for other elements of the same cluster. Prototypes combine the most representative values for the attributes of the elements that belong to the cluster. Consequently, prototypes are typical instances that represent the content of the cluster. They are calculated using an averaging function (in most of the cases, a weighted mean is used). However, when dealing with categorical sequences of events, those averaging functions are not applicable.

In [22] a new method for generating prototypes of event sequences was presented. This method is based on the calculation of an *Element Scoring Table (EST)*, where the elements can be either individual events or patterns of consecutive events considered as an indivisible unit. So, the method is called *Ordered Element Scoring Prototyping (OESP)*. Different scores are assigned to each event depending on its position in the sequences of the cluster. The ranking given by those scores is used to iteratively append an event to the prototype (initially the prototype is an empty sequence) until the prototype has a length similar to the average lengths of the sequences in the cluster.

This method will be used to calculate the centroids of the clusters obtained by the new similarity measure, which is needed to calculate the Sum of Square Error (3).

3 OSS_{OWA}: A new dissimilarity function for categorical sequences of events

From a formal point of view, a dissimilarity function f over two sequences of events s_1 and s_2 has to fulfill the following conditions:

1. Symmetry: $f(s_1, s_2) = f(s_2, s_1)$
2. Positivity: $f(s_1, s_2) \geq 0$ for any s_1 and s_2
3. Reflexivity: $f(s_1, s_1) = 0$

Some dissimilarity measures for sequences of symbols have been defined. For instance, Hamming distance [23] or Edit (Levenstein) distance [24] are usually considered [3]. However, such distances are quite simple when it is necessary to deal with the relative ordering positions of the symbols in the sequence, as it is needed in tracking studies such as the ones presented in the introduction. Another drawback is the difficulty of incorporating user (expert) preference policies in the computation of these measures.

As we are interested in comparing temporal event sequences, the results of those traditional categorical distances are not good since they disregard a lot of information that should be taken into account to obtain a more appropriate similarity value [12]. Two good examples of temporal event sequences are web logs and the itineraries of the tourists. In these scenarios, there are two key issues to be considered:

- The number of common elements inside the sequences.
- The order (position) among the common elements inside the sequences.

On one hand, the first consideration allows us to evaluate the common part of two different sequences, if this common part is large, then the two sequences have to be similar, because such sequences show that their owners have done the same things. On the other hand, the second consideration takes into account whether the events have been done in the same order or not. This second information is very useful when applying data mining processes. For example, when detecting common itineraries for planning new bus lines. Note that in this latter case, if two sequences have the same events placed in the same order, they have to be more similar than two sequences in which the events are placed in different order. Based on those two assumptions, in [12], the Ordered-based Sequence Similarity was defined (OSS).

However, in the context of event sequence comparison, apart from these two issues, it is also interesting to incorporate any user (expert) preferences on the similarity function. In some scenarios, it is more important to group sequences sharing a large number of events even though the events are not in the same order. On the contrary, in other scenarios, the user is interested in tracking the sub-sequences of common events, considering that two sequences with a few events in the same order are more similar than two sequences with a larger common set of unsorted events. Let us take as an example, an analysis of the tourist behaviour in a theme park. In the first case, the manager wants to find groups of visitors that have been at the same attractions (same locations) in spite of the order of visiting them. In the second case, the manager is interested in studying the flow of visitors in the park, and to know the common paths between attractions for planning or user-recommendation purposes.

The OWA operator with a fuzzy quantifier can be used to introduce the user aggregation policy into the computation of the similarity between pairs of temporal event sequences. This permits us to tackle the issues explained before. Based on this approach, a new similarity function can be defined, $d_{OSS-OWA}$. It is calculated in two steps.

Firstly, partial distances between the elements of the two sequences are calculated as follows.

Definition 3 *Let s and r be two event sequences. Let s_i be the i-th event of the sequence s, then the event distance of s_i with respect to r is defined as*

$$d(s_i, r) = \begin{cases} \frac{min_{s_i=r_j}|i-j|}{|s|+|r|} & if\ s_i \in r \\ 1 & otherwise \end{cases} \quad (4)$$

where $|s|$ and $|r|$ stand for the sequence lengths of sequences s and r respectively.

If s_i appears n times in s and m times in r, the occurrences in s and r are matched forming pairs that minimize $|i - j|$, if some occurrences do not have a couple ($n \neq m$), they are treated as if they do not appear in r, that is, they have a distance to r of 1.

Then, those partial distances are aggregated using the OWA operator, which permits to apply different aggregation policies[14].

Formally, given sequences s and r, we compute the event distances $d(s_i, r)$ for all $i = 1, \ldots, |s|$ and $d(r_i, s)$ for all $i = 1, \ldots, |r|$. Then, the OSS_{OWA} measure is defined as follows.

Definition 4 *Let s and r be two temporal event sequences with length $|s|$ and $|r|$, respectively, then the $d_{OSS-OWA}$ dissimilarity function of s and r is defined as*

$$d_{OSS-OWA(Q)}(s, r) =$$
$$OWA_Q(d(s_1, r), \ldots, d(s_{|s|}, r), d(r_1, s), \ldots, d(r_{|r|}, s))$$

where Q is a non-decreasing fuzzy quantifier, and $d(s_i, r)$ stands for the event distance of s_i with regards to r.

Following Definition 3, all event distances equal to 1 represents the case of two sequences with non-common events. On the contrary, distances smaller than 1 represent the shift between the common events in both sequences. In this case, the larger the event distance, the larger the shift. OWA operators permit the user to aggregate the values giving more importance to large or small values when appropriate quantifiers are selected. This stands for considering more important non-common events (large distance values) or common events (small distance values). In this way we are modeling the two scenarios described above.

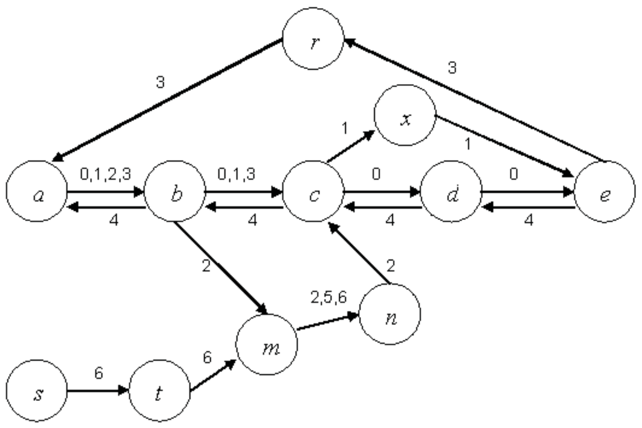

Figure 2: Example with 6 sequences.

Table 1: Original sequences.

id	Sequence
0	abcde
1	abcxe
2	abmnc
3	erabc
4	edcba
5	mn
6	stmn

4 Experiments

In this section, we present the results obtained with a synthetic sequences database (see Table 1). Figure 2 shows a graph representation of those sequences as paths between a set of 11 locations. These could be places inside a theme attractions park. The arrows indicate a transition from one attraction to the next one. The labels refer to the tourists that have followed each path step.

In order to test the $d_{OSS-OWA}$ function, the k-sized hierarchical clustering (KSHC) microaggreagtion algorithm has been used on this database. It has been applied to a k-anonymity value of $k = 2$, which means that cluster of 2 or 3 elements can be generated (recall that the number of elements is constrained by k and $2k-1$). Then, the method explained in section 2.3 for prototype generation has been performed to obtain the centroids of the clusters and calculate the information loss, that is, the SSE (3) .

Six different fuzzy quantifiers have been tested: $Q_1^{0.4}$, Q_1^1, Q_1^2, $Q_2^{0.1}$, $Q_2^{0.5}$ and $Q_2^{0.8}$. They define different aggregation policies in order to show how to introduce expert knowledge in the sequence similarity computation. Quantifiers with small α parameter consider more similar those sequences with a large set of common events. On the contrary, quantifiers with large α parameter consider more similar those sequences with few common events but placed in the same order. Note that, quantifiers with medium α parameter consider common events as important as events placed in the same order.

Table 2 shows the results obtained for the dataset given in Table 1. Each row corresponds to a different parametrization of the fuzzy measure used to compute the OSS_{OWA} dissimilarity function between pairs of sequences. The first three

Table 2: Microaggregated sequences with $k = 2$.

Q	SSE	Sequences
	0.515	abcxe, erabc
$Q_1^{0.4}$	0.767	abcde, edcba, abmnc
	0.907	mn, stmn
	0.032	erabc, edcba
Q_1^{2}	0.063	abmnc, abcde, abcxe
	0.081	mn, stmn
	0.002	erabc, edcba
Q_1^{4}	0.0	abcde, abcxe
	0.157	mn, abmnc, stmn
	0.934	abcxe, erabc
$Q_2^{0.1}$	1.137	abcde, edcba, abmnc
	1.614	mn, stmn
	0.087	erabc, edcba
$Q_2^{0.5}$	0.162	abmnc, abcde, abcxe
	0.159	mn, stmn
	0.002	erabc, edcba
$Q_2^{0.8}$	0.0	abcde, abcxe
	0.146	mn, abmnc, stmn

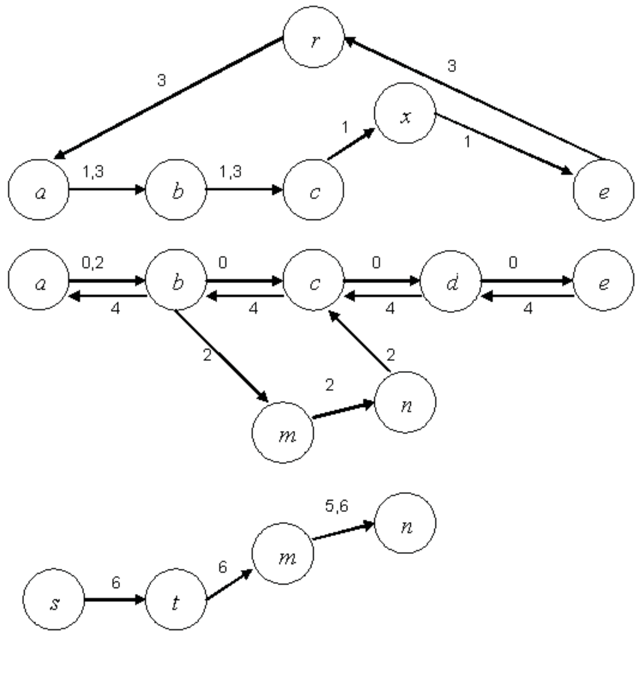

Figure 3: Clusters obtained with small α.

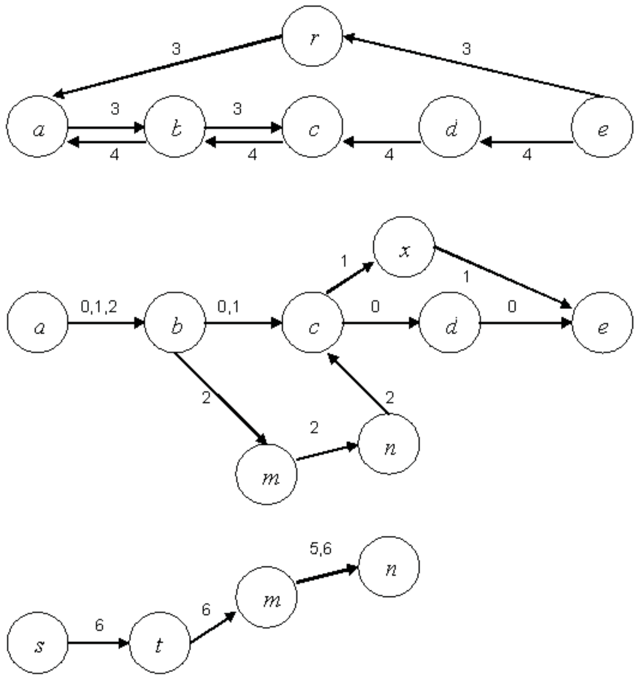

Figure 4: Clusters obtained with a medium α.

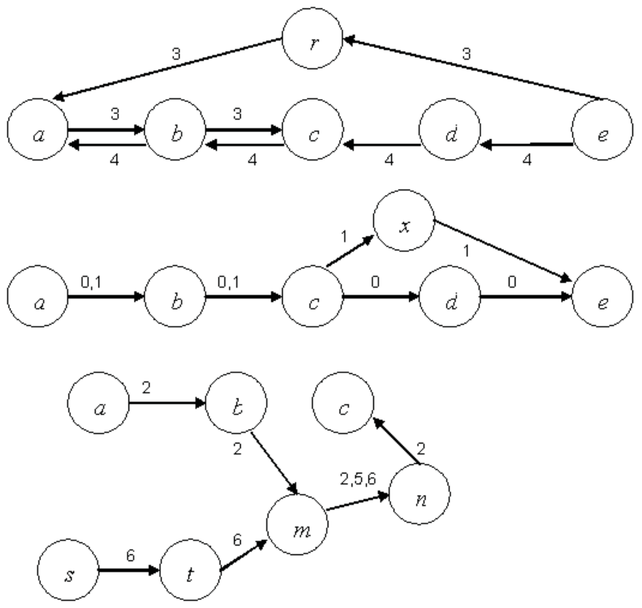

Figure 5: Clusters obtained with large α.

rows correspond to the fuzzy measure Q_1 (1) and the three last rows correspond to the fuzzy measure Q_2 (2).

The first thing to note is that the results for Q_1 and Q_2 are the same for low, medium and high α values. Using low values in the parameterization of the fuzzy measures (*e.g.* α equal to 0.4 for Q_1 and 0.1 for Q_2) generates a similarity function that stress the importance of visiting the same places, disregarding the event order. For this reason, the sequence $abcde$ is similar to the sequence $edcba$. In the same way, the sequence $abcxe$ is more similar to $erabc$ (with 4 equal elements) than to $abmnc$ (with only 3 common elements).

Oppositely, when large α values are selected (*e.g.* α equal to 4 for Q_1 and 0.8 for Q_2), the order among the common events plays a more important role in the similarity measurement. Consequently, the sequence $edcba$ is more similar to the sequence $erabc$ than to the sequence $abcde$, because the path between elements a , b and c is exactly the same. Note also that the sequences $abcxe$ and $abmnc$ also change their cluster with regards to the case of low α values.

To balance these two polices, a medium α value can be selected (*e.g.* α equal to 2 for Q_1 and 0.5 for Q_2). In this case, the groups obtained take into account both issues (common events and event order) in a similar way. Notice that in this case, the final groups are different from the ones in the other two cases.

The clusters obtained for the different α values are graphically displayed in Figures 3, 4 and 5, for small, medium and large α, respectively. From them, it can be more easily seen that the itineraries are grouped in a different manner depending on the aggregation policy.

To sum up, it can be said that the $d_{OSS-OWA}$ function permits to model different similarity measurement polices. This fact is crucial for data mining analysis and for privacy preserving data mining. If a microaggregation method does not consider the different semantics that the similarity function can have, the partition obtained may put together individuals that should be distinguished. Moreover, if the information of those individuals is then replaced by their prototype values, the posterior analysis of these masked data will not be able to retrieve some relevant knowledge for the user.

5 Conclusions and future work

In this paper, we have introduced a new similarity function for categorical event sequences using OWA operators. We have illustrated how it is possible to introduce expert knowledge inside the similarity computation using well-known fuzzy quantifiers. We have also presented some results about the application of this similarity function to a very recent microaggregation algorithm for categorical event sequences.

As future work we include the analysis of the microaggregation with respect to information loss and disclosure risk measures when it is applied to event sequences. These measures are required to properly evaluate the performance of the microaggregation when it is applied to event sequences and compare it with other approaches. Moreover, the effect of different fuzzy measures must be studied.

Acknowledgment

Partial support by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02), the Generalitat de Catalunya (grant 2005-SGR-00093) and Universitat Rovira i Virgili (project 2008TURISME-02). Jordi Nin wants to thank the Spanish Council for Scientific Research (CSIC) for his I3P grant.

References

[1] O. Abul, M. Atzori, F. Bonchi, and F. Giannotti. Hiding sequences. In *ICDE Workshops*, pages 147–156. IEEE Computer Society, 2007.

[2] N. Shoval, G. K. Auslander, T. Freytag, R. Landau, F. Oswald, U. Seidl, H.-W. Wahl, S. Werner, and J. Heinik. The use of advanced tracking technologies for the analysis of mobility in alzheimer's disease and related cognitive diseases. *BMC Geriatrics*, 8(7), 2008.

[3] G. Dong and J. Pei. *Sequence Data Mining*. Springer, 2007.

[4] T. G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30, London, UK, 2002. Springer-Verlag.

[5] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 2nd edition, 2006.

[6] L. Willenborg and T. de Waal. *Elements of Statistical Diclosure Control*, volume 155 of *Lecture Notes in Statistics*. Springer, 2001.

[7] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450, 2000.

[8] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. of Unc., Fuzz. and Knowledge Based Systems*, 10(5):571–588, 2002.

[9] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. of Unc., Fuzz. and Knowledge Based Systems*, 10(5):557–570, 2002.

[10] D. Defays and P. Nanopoulos. Panels of enterprises and confidentiality: The small aggregates method. In *Proceedings of 92th Symposium on Design and Analysis of Longitudinal Surveys*, pages 195–204. Statistics Canada, Ottawa, 1993.

[11] J. Domingo-Ferrer and V. Torra. A quantitative comparison of disclosure control methods for microdata. In *Confidentiality, disclosure, and data access: theory and practical applications for statistical agencies*, pages 111–133. Elsevier, 2001.

[12] C. Gómez-Alonso and A. Valls. *Lecture Notes on Artificial Intelligence: Modeling Decisions for Artificial Intelligence (MDAI 2008)*, chapter A similarity measure for sequences of categorical data based on the ordering of common elements, pages 134–145. Springer, 2008.

[13] V. Torra and Y. Narukawa. *Modeling decisions: information fusion and aggregation operators*. Springer, 2007.

[14] R. R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Trans. on Systems, Man and Cybernetics*, 18:183–190, 1988.

[15] R. R. Yager. Families of owa operators. *Fuzzy Sets and Systems*, 59:125–148, 1993.

[16] F. Felsö, J. Theeuwes, and G. Wagner. Disclosure limitation in use: Results of a survey. In *Confidentiality, disclosure, and data access: theory and practical applications for statistical agencies*, pages 17–42, 2001.

[17] S. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Trans. on Kwnoledge and Data Engineering*, 15(4):1043–1044, 2003.

[18] A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal United Nations Economic Commission for Europe*, 18(4):345–354, 2000.

[19] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Trans. on Kwnoledge and Data Engineering*, 14(1):189–201, 2002.

[20] A. K. Jain, M. N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[21] A. Valls and C. Gómez-Alonso. The k-sized hierarchical clustering method for microaggregation. Technical Report DEIM-RR-09-01, Universitat Rovira i Virgili, 2009.

[22] A. Valls, C. Gómez-Alonso, and V. Torra. Generation of prototypes for masking sequences of events. In *3rd Int. Workshop on Advances in Information Security, 4th Int. Conf. on Availability, Reliability and Security (ARES)*, pages 947–952. IEEE Computer Society, 2009.

[23] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[24] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.