The Application of Social Network Mining to Cattle Movement Analysis: Introducing the Predictive Trend Mining Framework

Puteri Nohuddin[1] and Frans Coenen[2] and Rob Christley[3]
[1]Institute of Visual Informatics, Universiti Kebangsaan Malaysia, Malaysia.
[2]Dept. of Computer Science, University of Liverpool, UK.
[3]Dept. of Epidemiology & Population Health, Institute Of Infection and Global Health, University of Liverpool, UK.
Email: puteri.ivi@ukm.edu.my, frans@liverpool.ac.uk, robc@liverpool.ac.uk

Abstract

This paper describes a predictive social network mining framework which is demonstrated using the Great Britain (GB) cattle movement datasets. The proposed framework, the Predictive Trend Mining Framework (PTMF), is used to analyse episodes of time stamped social network data. The PTMF has two main components (i) a Frequent Pattern Trend Analysis (FPTA) component that efficiently identifies temporal frequent patterns and trends and also provides a mechanism for clustering and analysing these patterns and trends so as to detect dynamic changes within the cattle movement network, and (ii) the predictive modelling component for forecasting the percolation of information or data across the network. The PTMF incorporates a number of novel elements including mechanisms to: (i) identify temporal frequent patterns and trends, (ii) cluster large sets of trends, (iii) analyse temporal clusters for pattern trend change detection, (iv) visualize these changes using pattern migration network maps and (v) predict the paths whereby information moves across the network over time.

Keywords: Social Network Mining, Trend Mining and Analysis, Prediction.

1. Introduction

Social network mining [1, 2, 3, 4, 5] has traditionally been focused on social www and mobile device applications such as Facebook, Twitter and LinkedIn. The objective of this focus was to identify influencers, sleepers, user clusters and so on, according to the nature and quantity of traffic across the network. However, these techniques have much wider application, for example with respect to co-authoring, logistics and customer networks. In this wider context the motivation is again to identify certain types of vertices and/or groups of vertices within the network, although the significance of these vertices will depend on the application domain. The work described in this paper is directed at social networks that describe the operation of the cattle industry, specifically that in operation in Great Britain (GB), where vertices represent holding areas such as farms, markets and abattoirs; and the edges connecting vertices representing the movement of "beasts" between these holding areas.

In the traditional setting social network mining is applied in the static context; a "snap shot" is taken of the network which is then processed off line. An alternative approach, and that of interest with respect to the work described in this paper, is to consider the network in the dynamic context with the aim of identifying communication (traffic) trends. This will then allow for the generation of prediction models. In the context of the global cattle rearing industry this will then provide information concerning seasonal cattle movement trends and allow policy makers to ascertain the effect of the introduction of new legislation and/or procedures.

The issue with social network trend mining is the large number of trends that are typically identified, which in turn will impede the analysis of the trend information extracted and the consequent generation of prediction models. In this paper the Predictive Trend Mining Framework (PTMF) is presented; a framework to support social network trend mining, analysis and prediction. The framework comprises

six stages: (i) trend identification (mining), (ii) trend grouping, (iii) pattern migration clustering, (iv) pattern visualization, (v) prediction modelling and (vi) prediction visualization. The first four stages collectively form the Frequent Pattern Trend Analysis (FPTA) component of the proposed framework. The work presented in this paper is a culmination of a body of work conducted by the authors in the context of trend identification and modelling with respect to social network data. Some of the elements of the PTMF have been previously presented individually, as have earlier variations of the work, of note are [6] and [7]; however, this is the only comprehensive presentation of the framework.

The rest of this paper is organized as follows. Some previous work in the domain of social network mining and trend identification is presented in Section 2. The motivational application domain, the UK cattle industry, is presented in Section 3; this will then be used to illustrate the operation of the PTMF with respect to the individual elements that make-up the framework. A formal definition of the problem domain is then given in Section 4. An overview of the PTMF is then presented in section 5 and in the following six sections (Sections 6 to 11) the individual elements are considered in more detail (illustrated using examples founded on the UK cattle industry application domain). The paper is then concluded in Section 12 with the presentation of a summary and some conclusions.

2. Previous Work

The work described in this paper, although directed at dynamic social network mining encompasses a number of different technologies including: pattern mining, trend mining, trend analysis and prediction modelling. Some of this work is well established and has an extensive literature associated with it, while other elements are less well studied. A brief overview of the technologies whose usage is advocated in this paper is presented in this section so as to provide the reader with some background context.

There has been a rapid increase in interest, within the data mining community, regarding the mining of social networks. Data mining based techniques are proving to be useful for the analysis of social network data, especially with respect to large datasets. Originally, social network mining approaches tended to be founded on graph mining techniques; but recently classification, clustering and link mining have also gained popularity. Typical social network mining applications include: (i) the discovery of information or disease spreading patterns from dynamic social interactions [8], (ii) the classifying of organizational users' roles in email distribution lists by using classification methods for predicting the role of a sender to the recipient of an email [9] and (iii) the prediction of product ratings from online customer networks for marketing purposes [10]. There are several case studies that describe the modification of traditional data mining methods for application to social network data. In [8], the authors investigated the effect of the overall network structure and designed an adaptable technique that combined rankings by various measures into a possible better list of "blockers" to prevent unhealthy spreading activities.

Social network mining can be applied in a static context, which ignores the temporal aspects of the network; or in a dynamic context, which takes temporal aspects into consideration. In the static context the analysis is directed at either: (i) finding patterns that exist across the network, (ii) clustering (grouping) sub-sets of the networks, or (iii) building classifiers to categorize nodes and links in a "snapshot" of the network. In the dynamic context, a different or extended kind of analysis can be applied to identify relationships between the nodes in the network by evaluating the spatio-temporal co-occurrences of events [11]. A central element of the work described in this paper is directed at mining social networks in the dynamic context, specifically in terms of the trends and change points that may exist within such networks.

As will become apparent, in the work described in this paper, trends are defined in terms of the frequency counts (support) associated with individual patterns. Given this definition, it is possible to identify some similarities with the work on Jumping and Emerging Pattern (JEP) mining (see for example [12]). Jumping patterns are usually defined as patterns whose support changes dramatically from one time stamp to another. Emerging patterns are then a special form of jumping pattern where the support changes from below some threshold σ to above σ, over two consecutive time stamps. The concept of frequent pattern trends defined in terms of sequences of frequency counts has also been adopted in [13]

in the context of longitudinal patient datasets. However in this case, unlike with respect to the work described in this paper, the trends were categorised according to pre-defined prototypes.

Trend analysis has been applied in many areas like healthcare [14], climate change [15] and human behaviour [8, 16]. Examples can also be found in the context of social network analysis. For example, in [17] a novel trend analysis algorithm was introduced to generate trends from web resources. The algorithm calculates the values of "temporal betweeness" of online social network node and link structures to observe and predict trends concerning the popularity of concepts and topics such as brands, movies and politicians. Likewise, some research directed at recommender systems [18, 19], and online market research [20], focuses on trends describing online social interactions and trusts so as to improve online marketing and sales strategies.

In data mining, prediction is a similar process as classification. However, prediction does not necessarily require a categorical class label to be included in the attribute set [21]. Prediction modelling is a significant analytical task concerned with predicting the probability of a future event or trend based on current and historical data. Neville and Frost [22] identified two types of social network modelling: descriptive and predictive modelling. Descriptive modelling views social relationships, in terms of network theory, as consisting of nodes and ties (edges, links, or connections), and also categories of social communities that may exist within a network. Whereas, predictive modelling is concerned with methods to analyse the changes in links or edges in a network, and also predict information interchange across a network. A number of studies have proposed techniques to predict how social network behaviour may change over time based on historical activity. Examples include: (i) the predicting and profiling of the behaviours of online bloggers for application in recommender systems [23], (ii) the prediction of relational entities and link relationships in networks using relational Markov Network algorithms [24], (iii) flu tracking using the content of Twitter [25] and (iv) the prediction of geographical location and friendships among Facebook users [26]. The work described in this paper proposes a prediction modelling technique to indicate the probability of particular events traveling across a network; for example how animal disease might spread across a network.

The framework proposed in this study incorporates many features that are lacking in much of the related work reviewed above. Furthermore, there is no specific study that compares clusters of social network data, which are related to "business communication data" patterns in the dynamic context and with respect to time series. Many proposed techniques feature inadequate visualization methods for representing the knowledge discovered from the social network data. The work presented in this paper consists of many components of work conducted by the authors with respect to trend pattern identification, modelling and visualizing so as to discovered useful knowledge with respect to social network data. The novel element presented in this paper are: (i) the unsupervised techniques adapted so that they can be used for the purpose of trend clustering to support the detection of changes in social network data in a specified time period and (ii) a trend cluster analysis mechanism so that trends and pattern migrations in the discovered clusters can be studied. Furthermore, the proposed visualization module provides for a useful mechanism whereby trend pattern migrations, from one trend cluster to another over a period of time, can be displayed. This visualisation benefits users, practitioners and decision makers alike, so as to support and contribute to important decision making processes.

3. Motivational Example

The focus for the work described in this paper is the operation of the GB cattle mining industry as encapsulated by GB Cattle Tracking System (CTS) which records all movements of cattle registered within or imported into GB. The database is maintained by the Department for Environment, Food and Rural Affairs (DEFRA), the government department in the UK concerned with policy and legislation in areas such as farming, and animal health and welfare. The CTS was introduced in September 1998, and updated in 2001 to support disease control activities. In 2012, the CTS database held some 155 GBytes of data. Table 1 describes some characteristics of CTS database.

Table 1: CTS database

Date:	2003-2006.
Size:	About 155GB.
Significant tables:	Animal, location and movement.
Data records:	More than 1 million records; each record describes an animal movement
Attributes:	A time stamp (month and year); the number; gender; age; category (dairy, meat, etc.) and breed of the cattle moved; the sender "type" (farm, market, abattoir, etc.) and geographical location in terms of easting and northing grid values; and the receiver location "type", and easting and northing grid values.

The CTS database comprises a number of tables, the most significant of which are the animal, location and movement tables. For the analysis incorporated into this paper the data from 2003 to 2006 was extracted. The CTS dataset comprises one record per beast moved, over 1,000,000 per year. These records were conflated by combing records that occurred at the same time between the same sender and receiver locations. This reduced the number of records to be considered to about 400,000 per year. Each composite record comprised: (i) a time stamp (month and year); (ii) the number, gender, age, category (dairy or meat) and breed of the cattle moved; (iii) the sender "type" (farm, market, abattoir, etc.) and geographical location in terms of easting and northing grid values; and (iv) the receiver location "type" and geographical location in terms easting and northing grid values. Note that if two different breeds of cattle were moved at the same time from the same sender location to the same receiver location, this would generate two records. It is also interesting to note that the maximum number of cattle (of the same breed) moved between any pair of locations for a single time stamp was approximately 40. The ground area covered by the CTS database was divided into a k × k grid. These grid squares were then conceptualized as being the vertices within the context of a social network representation. Edges between grid squares then represented cattle movement between grid squares. The authors conducted experiments using various grid sizes, but k = 50 and k = 100 were found to be the most appropriate.

To permit the desired processing and analysis the datasets were translated into a binary valued attribute format whereby nominal data items were assigned unique attribute labels and numeric data items were ranged and each sub-range assigned to a unique attribute label. The overall number of attributes was dependent on the value of k used; but typically some 450 attributes were used, including 185 attributes representing different breeds (some quite rare), 14 different sender types and 14 different receiver types.

4. Formalism

From the above the input to the Frequent Pattern Trend Analysis (FPTA) component of the framework comprises a sequence of n time stamped datasets, $D = \{d_1, d_2, ..., d_n\}$. So that comparisons between identified trends could be conducted the data was subdivided into *e* episodes, each of equal length *m*, thus $n = e \times m$. However, with respect to the CTS database a granularity of one month seemed like an obvious choice hence m was set to 12; consequently each episode represented a year (for example, the GB cattle movement data was divided into four episodes: 2003, 2004, 2005 and 2006). Thus a total of 48 (4×12) individual social networks were used to represent the CTS application.

From the foregoing the data was discretized into a binary valued format (the reason for this will become apparent in Section 6). Each record in each dataset d_i is thus defined by a global set of attributes $A = \{a_1, a_2, \ldots, a_m\}$ each of which could take the values 0 or 1 (exist or does not exist), however only positive values are recorded in the discretised datasets (absence implies a 0 value). Each record thus comprised some subset of A. Note that the number of records in each dataset need not be constant across the collection.

A trend is a sequence of values, associated with some pattern, arranged in some chronological order. With respect to the PTMF the patterns of interest were defined to be frequently occurring item sets, subsets of A that occurred frequently across a dataset; an idea founded on the concept of frequent item set and association rule mining (Agrawal et. al 1993, Agrawal and Srikant 1994). Thus a frequent item set I is some subset of A that occurs frequently in a dataset d_i. The frequency of an item set A within d_i is determined by a "support count" s of the number of times I appears in d_i, if s is greater than some support threshold σ I is deemed to be a "frequent item set". In the context of the PTMF a trend is then a chronological sequence of support counts. Thus, a trend line t, for a particular pattern I, comprises a set of values $t_I = \{s_1, s_2, \ldots, s_m\}$ where each value represents an occurrence count of the pattern I at a particular time stamp within an episode of length m. The collection of trend lines for a pattern I, T_I, is then comprised of a sequence of trend lines (one per episode) $\{t_{I_1}, t_{I_2}, \ldots, t_{I_e}\}$ (where e is the number of episodes). The entire collection of trends within a system is given by τ. The trend lines associated with a time stamp i is given by τ_i, thus $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. The entire collection of trends associated with a particular episode i is given by τ_{e_i}. Thus $\tau = \{\tau_{e_1}, \tau_{e_2}, \ldots, \tau_{e_e}\}$. The trend lines associated with a specific episode e_i are given by $\tau_{e_i} = \{\tau_k, \tau_{k+1}, \ldots, \tau_{k+m}\}$, where k is the first time stamp in the episode e_i.

5. PTMF Overview

Figure 1 gives an overview of the PTMF. From the figure it can be seen that the elements that make up the PTMF have been divided into two groups: predictive modelling and FPTA. The input (middle left in the figure) to the PTMF is a collection of networks divided into episodes, 4×12 in the case of the CTS application. The first stage is then trend identification founded on the concept of frequent item set mining; an algorithm specifically developed for this purpose, called Trend Mining-Total from Partial (TM-TFP), is suggested.

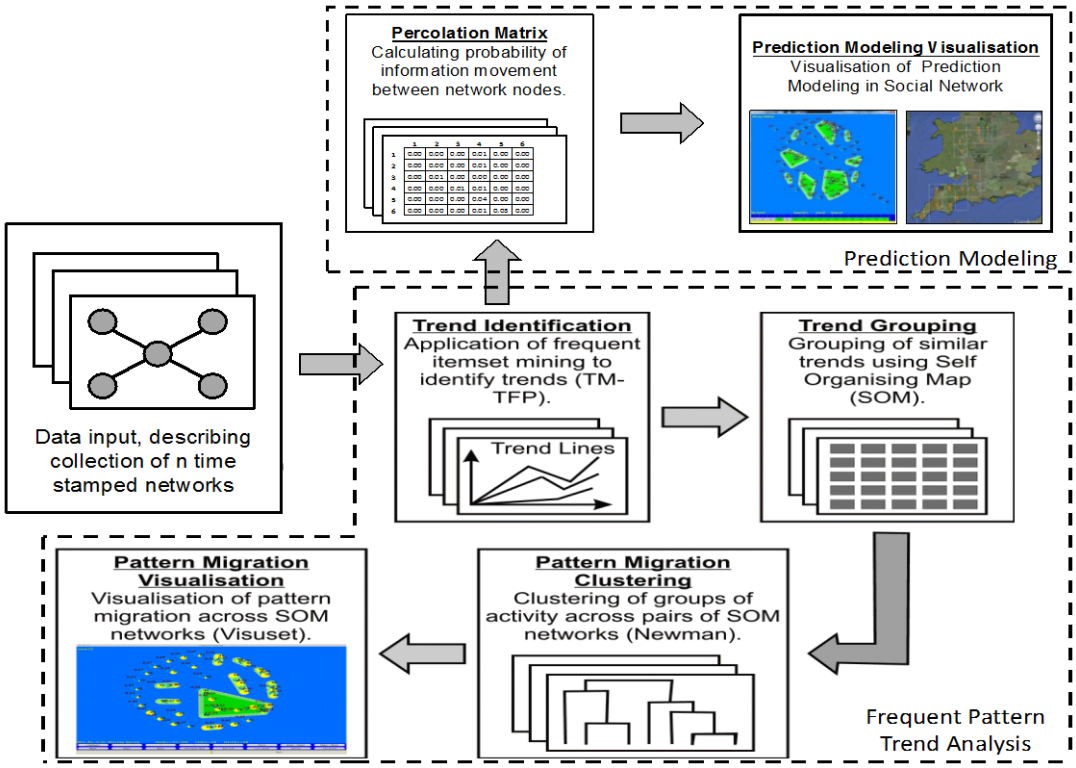

Fig. 1: *Schematic Illustrating the Operation of the PTMF*

With respect to frequent pattern trend analysis the next stage is trend grouping. The idea is to facilitate analysis by clustering similar trends; this was achieved using Self Organising Map (SOM) technology. To analyse how trends change between episodes the following stage was to compare SOM maps by identifying significant trend migrations from one trend type (SOM node) to another. This was achieved

by generating a migration map and applying a hierarchical clustering mechanism; in this paper we advocate Newman hierarchical clustering but other similar algorithms will suffice. The final stage (bottom left in the Fig. 1) is a mechanism for providing a visualisation of the migrations, to provide for additional analytical support.

In the context of prediction modelling the identified trend lines are used to predict future (cattle) movements through the generation of a v × v "percolation matrix", where v is the number of vertices in a given network. With respect to the CTS application we are interested in predicting cattle movements, for example with a view to predicting the spread of bovine diseases. The predictions can then be visualised using either a geographical map or a representative map (top right in Fig. 1).

Each of the six stages making up the PTMF is considered in further detail in the following six sections.

6. Trend Mining

As already noted the patterns of interest are frequent item sets as popularised in Association Rule Mining [27, 28]. To mine patterns and trends an extended version of the Total from Partial (TFP) frequent pattern mining algorithm [29, 30], called TM-TFP, was adopted. TFP was selected because it was an established frequent pattern mining algorithm and because of its efficiency derived from the use of two set enumeration trees, the Partial (P) tree and the Total (T) tree. The distinction between TFP and TM-TFP is that the latter uses an additional TM-tree to integrate T-trees associated with individual time stamps. The TM-tree has a similar structure to a T-tree. Each node in a TM-tree has two fields: (i) trend and (ii) reference. The first field consists of a vector, of length m, holding support values that describe the associated trend line. The second holds a reference (pointer) to the next level of the TM-tree. As in the case of the T-tree, the individual item labels making up the frequent patterns are not explicitly stored as this information can be obtained directly from the tree structure. The TM-tree also has a TM-tree header which holds references to a set of e T-trees (recall that e is the number of episodes in a given system). The TM-TFP trend mining algorithm is presented in more detail below.

An overview of the operation of the TM-TFP algorithm is given by the pseudo code presented in Algorithm 1. The algorithm commences by initialising the TM- tree header according to the number of time stamped datasets held in D (line 1). Then, lines 2 to 5, it loops through the datasets d_1 to d_n contained in D and creates individual P-trees and T-trees, one per dataset d_i, using the TFP algorithm. Note that a reference to each generated T-tree is stored in the TM-tree header (line 5). Finally, line 7, the buildTMTtree() method is called to process the collection of T-trees built from $\{d_1, d_2, \ldots, d_n\}$, and construct the desired TM-tree.

```
Input : A set of n datasets, D = {d1, d2, . . . dn}, σ
Output: TM-tree holding frequent item set trends found in D

//  Initialise TM-Tree header file
1) TM-tree Header size ← n;
2) For i←0 to (n-1) do
3)     Create P-treei from di;
4)     Create T-treei, using σ;
5)     TM-tree_Header [i] ← T-treei reference;
6) end
7) buildTM_Ttree() // Algortihm 2
```

Algorithm 1: *TM-TFP algorithm for building TM-tree*

Algorithm 2 describes the buildTMTtree() method. The algorithm commences (line 1) by determining the total number of frequent item sets that are required to be held in the TM-trees. The TM-tree is then initialised using this number so that it can hold all combinations of frequent item sets and trends contained in the n T-trees. The TM-tree is then constructed (lines 2 to 4) by repeated calls to the

buildTMTtree method. The pseudo code for this method is presented in Algorithm 3. The input to this algorithm is the current T-tree, T-tree$_i$, and the current time stamp i. Note that on line 3 of Algorithm 2 there is a test for the empty T-tree situation (which may exist if a very high support threshold is used).

```
Input : A set of n T-trees
Output: TM-tree holding frequent item sets in D

// builds TM-tree by processing collection of T-trees associated with Dn
1)  Initialise TM-tree to accommodate all n T-trees;
2) for i←0 to (n-1) do
3)      if T-treei ≠ ∅ then
4)          buildTM_Ttree (T-treei,i);    // Algorithm 3
5)      end
6) end
```

Algorithm 2: *buildTM_Ttree()*

Algorithm 3 proceeds by processing the top level of the given T-tree (T-Tree$_i$). For each element in this top level, T-tree$_i$[k], if the element is not empty the algorithm calls (line 3) the addToTMtree method (Algorithm 5) with the associated support. If T-tree$_i$[k] has child nodes, these are then processed (Algorithm 4). If T-tree$_i$[k] has no child nodes (is not supported) the algorithm calls (line 8) the addToTMtree method, with a support value of 0.

```
Input : T-treei, time stamp i
Output: TM-tree holding frequent item sets in D

// loop through top level of current T-tree
1) for k ← 1 to numOfoneItem sets do
2)      if T-treei[k] ≠ ∅ then
3)          addToTMtree(T-treei[k].support, i);    // Algorithm 5
        // move down a level
4)          if T-treei[k] has child node then
5)                  buildTM_Ttree(T-treei[k].child, i,k-1);   // Algorithm 4
6)          end
7)      else
8)          addToTMtree(0, i);                              // Algorithm 5
9)      end
10) end
```

Algorithm 3: *buildTM_Ttree(T-treei,i)*

Algorithm 4 processes the child nodes of a given T-tree$_i$. The size parameter is the number of elements at the current node in the current branch of the given T-tree. The algorithm loops through the elements in the current level of the tree (line 1). If the node represented by an element is not empty (i.e. it represents a supported item) the equivalent node in the TM tree is updated with the support count (Algorithm 5). In line 4 the algorithm tests whether the current node (element) has a child branch associated with it. If so this next level is processed by a repeat call to Algorithm 4, and so on until there are no more child branches to be processed. The process of building the TM-tree continues in this manner until all T-trees have been considered. Line 8 in Algorithm 4 deals with the situation when a T-tree level element is not supported, in this case a call is again made to the addToTMtree method with a support value of 0.

```
Input :Ttreei, time stamp i, size
Output: TM-tree holding frequent item sets in D

1) for k ← 1 to size (of current node level of the current T-tree) do
```

```
2)    if T-tree_i[k] ≠ ∅ then
3)        addToTMtree(T-tree_i[k].support, i);          // Algorithm 5
// move down a level
4)        if T-tree_i[k] has child node then
5)            buildTM_Ttree(T-tree_i[k].child,i,k-1);   // Algorithm 4
6)        end
7)    else
8)        addToTMtree(0, i);
9)    end
10) end
```

Algorithm 4: *buildTMTtree(T-treei,i,size)*

Algorithm 5 adds the support value of an item set in a given T-tree$_i$ to the TM-tree. If the given T-tree$_i$ element (node) representing the item set is empty (the item set is not supported) the support value will be 0. The value zero is used as a flag to indicate that an item set is not supported. Whatever the case the added support for the item set will form part of the eventual trend line, stored at the TM-tree node, for the item set.

```
Input : support, timestamp i
Output: TM-tree updated with item set support

1)  TM-tree.trend[i] = T-tree_i.support;
2)   return;
```

Algorithm 5: *addToTMtree(support, i)*

The number of trend lines identified using the TM-TFP algorithm depends on the nature of the σ value used. A trend line is recorded if it is frequent (has support greater than σ) on at least one occasion, otherwise it is not recorded. Table 2 gives the number of trends identified using TM-TFP using a range of σ values (0.5%, 0.8%, 1.0%); it is common practice in frequent item set mining to use low values for σ so as to ensure no interesting pattern is missed. From the table, it can be seen that a large number of trends are identified and (as might be expected) that the number of frequent patterns discovered increases as the value for σ decreases. Table 3 shows some comparative run time values so as to give an indication of the time complexity of the TM-TFP algorithm. From the table it can be seen that increases in σ results in corresponding linear decreases in the TM-TFP run time (because fewer patterns are discovered).

Table 2: Number of trends identified using TM-TFP for a sequence of four CTS network episodes and a range of support thresholds

Episode	Support Threshold		
(year)	0.5%	0.8%	1.00%
2003	63,117	34,858	25,738
2004	66,870	36,489	27,055
2005	65,154	35,626	25,954
2006	62,713	33,795	24,740
Average	64,464	35,192	25,872

Table 3: The TM-TFP algorithm run time values (seconds) using the CTS social network episodes

Episode	Support Threshold		
(year)	0.5%	0.8%	1.00%
2003	97.02	69.49	63.54
2004	92.44	70.00	64.00
2005	83.25	59.55	53.50
2006	101.06	72.95	66.09
Average	93.44	68.00	74.16

Some examples of frequent patterns generated from the CTS network data are presented in Table 4. The patterns include information on: animal age, animal gender, breed name, breed type (dairy or beef), number of cattle moved, and sender and receiver location type and grid square area. It should be noted that trend lines 1 and 2, although associated with different patterns, are very similar in shape; this was found to be a common feature of this type of network analysis and lead to the idea of trend grouping to

facilitate further analysis (this is discussed in the following section).

Table 4: Examples of frequent patterns and associated trends obtained from the 2004 CTS network using σ=0.5%

No.	Pattern	Trend Values
1.	{Sender PTI=4, Receiver Location type = Slaughterhouse (Red Meat), Sender Location type = Agricultural Holding, age <= 1 year, gender = male}	{1432, 1485, 1712, 1451, 1334, 1350, 1473, 1716, 1816, 1627, 1751, 1640}
2.	{Sender PTI=4, Receiver Location type = Slaughterhouse (Red Meat), Sender Location type = Agricultural Holding}	{3561, 3308, 3676, 3072, 2944, 2958, 2927, 2966, 2945, 2276, 2798, 2754}
3.	{Sender PTI=4, Receiver Location type = Slaughterhouse (Red Meat), Sender Location type = Agricultural Holding}	{4540, 4329, 4959, 4220, 4302, 4842, 4431, 4414, 4290, 4514, 5593, 5590}
4.	{Receiver PTI=4, Breedtype =beef, Breed =Limousin, gender = male, }	{1699, 1631, 2369, 2637, 2711, 2177, 1947, 2099, 1994, 2359, 2021, 1338}
5.	{Number Cattle moved <= 5, Breedtype =beef, Breed =Belgian Blue Cross}	{1163, 1106, 1348, 1273, 1203, 1107, 1119, 1184, 1289, 0, 1474, 1259}

7. Trend Grouping

From Table 2 it can be seen that a considerable number of trends may be identified, so many that it is difficult to see how this knowledge might be useful to decision and policy makers. However, many of the trends display similar characteristics, this observation gave rise to the idea of grouping trends. There are various ways that the desired grouping could be undertaken and the authors considered a variety of clustering options; however, it was quickly realized the Self Organising Map (SOM) technology provided the most appropriate solution. Likewise, SOM is desirable because, in addition to its capability to cluster a large number of trends, the discovered clusters can be presented in a trend line to illustrate types of trends that are discovered in the processed data. SOMs were first introduced in [31, 32]. Fundamentally, a SOM may be viewed as a neural network based technique designed to reduce the number of data dimensions in some input space by projecting it onto a n×m "node map" which groups similar data items together at the nodes. The SOM learning process is unsupervised; nevertheless, the n×m number of nodes in the SOM must be prespecified. Currently, there is no scientific method for determining the best values for n and m. However, the n × m value does define a maximum number of clusters; although on completion some nodes may be empty [33].

To train a SOM requires input data D and a set of weight vectors (W), one per input attribute, to which distance and neighbourhood functions are applied to determine which nodes the records in D should be associated with. Different weight distributions will be associated with different nodes. Initially the weight vectors are initialized randomly. Then, the algorithm processes the input data, record by record. For each record, each node "bids" for the record and the record is assigned to the "winning" node. This is done using a distance function. As a result the record is assigned to a node and the neighbouring weights are adjusted to reflect the nature of the most recently assigned record. At first the adjustments are relatively large, but as the training continues the adjustments become smaller and smaller. A feature of the adjustment is that adjacent nodes come to hold similar records; the greatest dissimilarity is therefore between nodes at opposite corners of the map. The process continues in an iterative manner continuing to tune the map until all records in the training set have been allocated.

In the context of the CTS application one SOM was generated for each episode. Experiments were conducted using different sizes maps as a result of which a 10×10 node SOM was found to be the most

effective as this gave a good decomposition while still ensuring computational tractability. The input to the SOM algorithm was therefore a set of trend lines and a set of weight vectors of length 12; because each trend line was described in terms of 12 support values (trend values). The SOM was "trained" using the data associated with the first 2003 episode (τ_{e_1}). The process is described by Algorithm 6. With reference to this algorithm the SOM was first initialised (line 1) with a predefined x × y grid (map). The training then takes place in lines 3 to 5. Euclidean distance and Gaussian neigbourhood functions were used to determine the distance values between data and the "winning" node and adjust weight vectors of nearby map nodes accordingly. At the end of the training phase a "prototype" map was produced that represented the types of trend lines that existed within τ_{e_1}. Copies of the resulting prototype map were then populated with data from the all the episodes (τ_{e_1} to τ_{e_e}), to produce a sequence of e maps M = {M_1, M_2, ..., M_e} (line 8 to 12 in the algorithm).

```
Input :T={T_e1, T_e2, ..., T_ee}
Output: SOM prototype map and n trend line maps

// generate a SOM prototype map
1)   Initialise a SOM prototype map with x×y nodes;
2) Assign weight vectors, w, to the map nodes;
3) for i←0 to | τ e1| do
4)        Find the "winning" node for trend line t1 i in the prototype map;
5)        Adjust the weight vectors of nearby map nodes accordingly;
6) end
//  generate a SOM trend line maps
7) for k←0 to e do
8)          Initialise a SOM trend line map, with x × y nodes for episode k;
9)       for i ← 0 to |T_ek | do
10)            Locate t_ki onto the prototype map for episode k;
11) end
12) end
```

Algorithm 6 *Trend Grouping using SOM*

An example prototype map is presented in Figure 2. The example was generated from episode τ_{e_1} (2004) using a 7×7 map configuration (although a 10×10 map is more desirable a 7×7 map has been used here for illustrative purposes as it is simpler to interpret). For each node the X-axis records the month identifier and the Y-axis the support value. Figure 3 shows the prototype map from Figure 2 once it has been populated with data from τ_{e_1} (the same episode used to train it, but a similar result would be produce using any of the other episodes). A colour coding is used in Figure 3, the darker the trend lines in a node the more trend lines that have been allocated to that node.

8. Pattern Migration Analysis

To further aid the analysis of the data it was considered desirable to identify "trend migrations". A trend migration was defined as how a trend line associated with a particular pattern might change (or not change) over time. In other words whether a trend line t remained in the same location (node) with respect to successive SOMs M_e and M_{e+1} or moved (migrated) from one location to another. The motivation here was that, with respect to the CTS network data, discussion with potential end users indicated that they would be interested in how trends associated with particular patterns changed over time.

To this end the correlations between a pair of SOMs were interpreted as a lattice, a "migration map", with every node potentially connected to every other node. In the same way that the nodes in (say) Facebook or Twitter social networks may be potentially connected to every other node. A pair of SOM nodes was deemed to be strongly connected if they featured significant migration between M_i and M_{i+1}. Thus, for each sequential pair of i×j SOMs, M_e and M_{e+1}, a "migration map" was constructed comprising i×j nodes and potentially $(i \times j)^2$ links (including "self links"). The nodes in the map were labelled with

the number of patterns held in the corresponding nodes in SOM map M_e (the from map). The links then represented the migration of patterns from M_e to M_{e+1}, and were labelled with the number of migrating patterns (thus a "traffic" value); the higher the value the stronger the link. Two mechanisms for analysing these migrations were proposed; the first was founded on distance travelled and the second on volume (number of similar migrations).

With respect to distance travelled a Euclidean distance function was used. This "travel distance" can then be used to determine the significance of individual migrations; the greater the distance the more significant the change. Thus, given a sequence of trend line SOM maps, comparisons can be made to see how trends associated with individual frequent patterns change by considering the nodes in which they appear and disappear over time. The distance travelled pattern migration analysis pseudo code is presented in Algorithm 7. The algorithm commences by defining a |FP|×e table (where FP is the set of all identified frequent patterns). The rows in the table represent individual frequent patterns and the columns the individual SOM maps, one per episode (thus four columns in the case of the CTS data). The table was then populated with the SOM node IDs associated each discovered frequent pattern for all e SOM maps $M = \{M_1, M_2, ..., M_e\}$ (line 4). Then, in line 7, the algorithm defines a sequence of e−1 "Migration Matrices" (MMs) for each sequential pair of SOMs. Each matrix measured x×x, where x is the number of nodes in any individual SOM (x=i×j). An example migration matrix is presented in Table 5. The migration matrix shows the numbers of patterns that have migrated from M_e to M_{e+1}. In Table 5 $n_{i,i}$ is the number of patterns that have stayed in node c_i in both SOMs (the term "self-link" is used to indicate such migrations), $n_{i,j}$ gives the number of patterns that have migrated from node c_i in M_e to c_j in M_{e+1}. Returning to Algorithm 7 the process is completed (lines 8 and 10) with the population of the migration matrix.

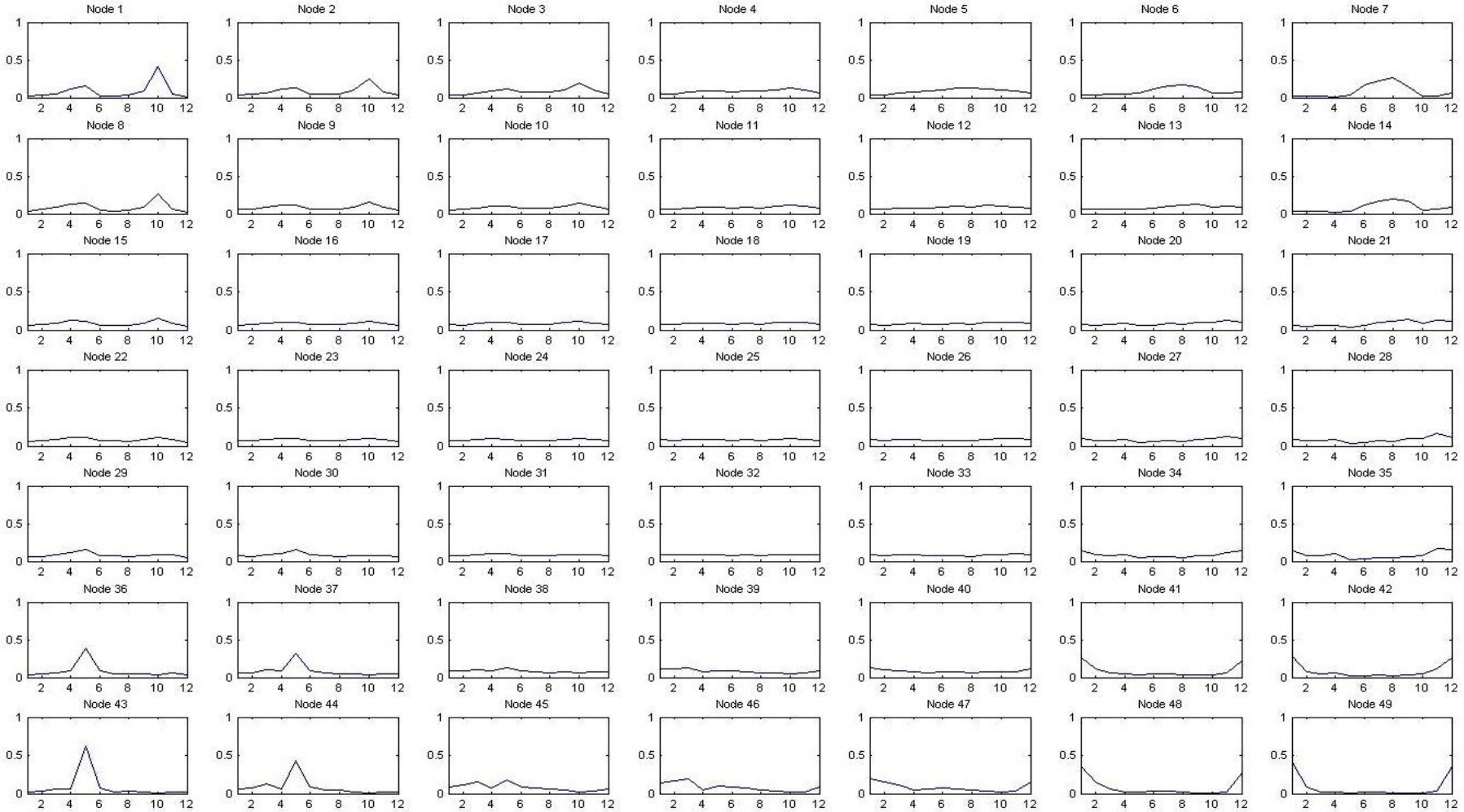

Fig. 2: *Prototype map for τ_{e_1} (2003) using a 7×7 node map*

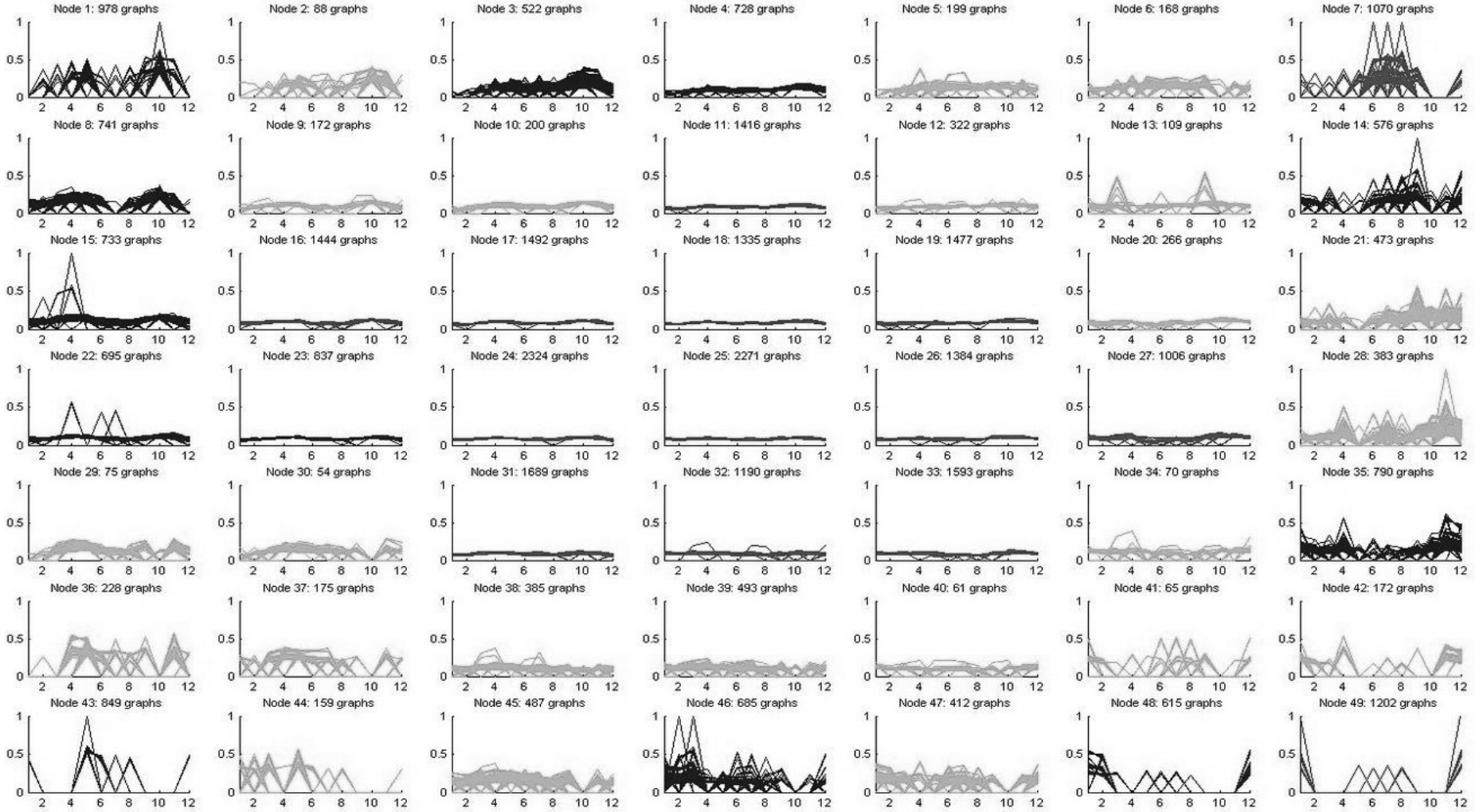

Fig. 3: *Trend line map for τ_{e_1} (2003) generated using the prototype map given in Figure 2.*

```
Input : FP = Set of all frequent patterns identified in episodes {e1, e2, …, ee}
Output: Sequence of (e−1) Migration Matrices
1) Define Table measuring |FP|×e;
2) for i ← 1 to e (step through episodes) do
3)         for j←1to |FP|(step through these to FP) do
4)                 Table[i][j] = Table[i][j] ∪ SOM node ID for ei;
5)         end;
6) end;
7) Define(e−1)Migration Matrices(MMs),each measuring(x×x)where x is the
   number of SOM nodes;
8) for i←1 to (e−1) do
9)         for k←1to |FP| do
10)                Increment count at MMi[Table[K][i]][Table[K][i+1]];
11)        end;
12) end;
```

Algorithm 7: *Distance travelled pattern migration analysis*

Table 5: Format of a Migration Matrix

	C_1	C_2	C_3	C_4	…	C_5
C_1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$	$N_{1,4}$	…	$N_{1,n}$
C_2	$N_{2,1}$	$N_{2,2}$	$N_{2,3}$	$N_{2,4}$	…	$N_{2,n}$
C_3	$N_{3,1}$	$N_{3,2}$	$N_{3,3}$	$N_{3,4}$	…	$N_{3,n}$
C_4	$N_{4,1}$	$N_{4,2}$	$N_{4,3}$	$N_{4,4}$	…	$N_{4,n}$
…	…	…	…	…	…	…
C_n	$N_{n,1}$	$N_{n,2}$	$N_{n,3}$	$N_{n,4}$	…	$N_{n,n}$

Some examples of CTS patterns migrations, from one node to another, between the SOMs for the episodes 2003 to 2006 are shown in Table 6. Thus, from the table, it can be seen that, for example, the trend line associated with the pattern {Receiver Area = 14, Sender Area = 13, Animal Age ≤ 1 year old} was in node 10 in the 2003 SOM map and moved to node 8 in the 2004 SOM map, then to node 44 in the 2005 SOM map ending up in node 18 in the map for 2006. The pattern {Number cattle moved ≤ 5, Receiver location type = Slaughter House (Red Meat), Receiver Area = 14, Sender Area = 13, Gender = female} was only considered significant (frequent) in two data episodes, 2004 and 2005. It was in node 20 in the SOM map for 2004 and moved to node 80 in the map for 2005. Referring to the associated CTS prototypes it can be seen that the change of trend "type", from trends featuring high support between September and December 2004 to trends that feature high support between December and January 2005, could be said to be significant (or at least interesting).

Table 6: Example of migrating CTS Frequent Patterns trends

No.	Pattern	2003 SOM Node	Dist	2004 SOM Node	Dist	2005 SOM Node	Dist	2006 SOM Node
1	{Receiver Area = 14, Sender Area = 13, Animal Age ≤ 1yearold}	10	2.0	8	5.1	44	4.8	18
2	{Number cattle moved ≤ 5, Receiver location type = Slaughter House (Red M eat), Receiver Area = 14, Sender	0	0.0	20	9.8	80	0.0	0

	Area = 13, Gender = female}							
3	{Receiver Area = 44, Sender Area = 44, Breed Type = Beef, Breed = Belgian Blue Cross}	6	1.0	5	2.8	21	5.0	38
4	{Receiver Area = 35, Sender Area = 35, Breed Type = Beef and Dairy, Animal Age ≤ 1yearold}	7	6.7	46	6.3	6	0.0	0
5	{Receiver PTI = 4, Receiver Area = 14, Sender Area = 13, Gender = female}	33	1.4	27	5.1	29	0.0	39

With respect to migration volume it was deemed desirable to identify frequent migrations, migrations associated with a significant number of patterns whose associated trend lines were altering in the same way (this in turn might be indicative of a change in working practices or the effect of a policy change). In other words we wish to identify vertices in migration maps that were strongly connected (collectively featured a significant amount of migration). Work on social network mining had already suggested ways of identifying strongly connected clusters of nodes [33, 34]. With respect to the work described here, and as noted above, an agglomerative hierarchical clustering mechanism, founded on the Newman method [35] for identifying clusters in network data, was adopted. Newman proceeds in the standard iterative manner on which agglomerative hierarchical clustering algorithms are founded. The process starts with a number of clusters equivalent to the number of nodes. The two clusters (nodes) with the greatest "similarity" are then combined to form a merged cluster. The process continues until a "best" cluster configuration is arrived at or all nodes are merged into a single cluster. Best similarity is defined in terms of modularity value Q calculated as follows:

$$Q_i = \sum_{i=1}^{i=n} (c_{ii} - a_i^2)$$

where Q_i is the Q-value associated with the current cluster i, n is the total number of nodes in the network, c_{ii} is the fraction of intra-cluster (within cluster) links in cluster i over the total number of links in the network, and a^2_i is the fraction of links that end in the nodes in cluster i if the edges were attached at random. The value a_i is calculated as follows:

$$a_i = \sum_{j=1}^{j=n} c_{ji}$$

where c_{ij} is the fraction of inter-cluster links, between the current cluster i and the cluster j, over the total number of links in the network. The values for c_{ii} and c_{ij} were extracted from a previously generated migration matrix (Table 3).

Hence, on each iteration, the Q-values for all possible cluster pairings are calculated and the pairing with the highest Q-value selected for merging. The process proceeds until a best cluster configuration is achieved. This is defined as the configuration with the highest overall Q-value. According to (Newman and Girvan 2005) communities can be said to exist within a network if the Q-value is above 0.3. Note that if all nodes are placed in one group the Q-value will be 0.0 (i.e. a very poor clustering). Thus, using Newman clustering, we can identify groups of SOM nodes that feature significant migration. The outcome of the pattern migration analysis in terms of volumes (the identification of migration communities) is considered further at the end of the following section.

9. Pattern Visualisation and Animation

In the context of the migration maps identified above, and the communities identified using Newman hierarchical clustering, it was also considered desirable to provide mechanisms whereby the migrations can be observed. Two forms of visualisation were considered: (i) visualisation of pattern migrations between two successive SOMs and (ii) animation of the pattern migrations between three successive SOMs. In each case, the migration maps also illustrated the discovered pattern migration communities. The communities were depicted as "islands" separated by a "shoreline". The visualisation/animation process was founded on an extension of the Visuset software system [37] and is described in this section.

For the visualisation, Visuset locates nodes in a 2-D "drawing area" using the Spring Model [38]. The spring model for drawing graphs in 2-D space is designed to locate nodes in the space in a manner that is both aesthetically pleasing and limits the number of edges that cross over one another. The migration map to be depicted is conceptualized in terms of a physical system where the edges represent springs and the nodes objects connected by springs. Nodes connected by "strong springs" therefore attract one another while nodes connected by "weak springs" repulse one another. The maps are drawn following an iterative process. Nodes are initially located within the 2-D space using some set of (random) default locations (usually defined in terms of an x and y coordinate system) and, as the process proceeds, pairs of nodes connected by strong springs are "pulled" together. In the context of the CTS application the spring value was defined in terms of a correlation coefficient (C):

$$C_{ij} = \frac{X}{\sqrt{(|M_{e_k}i| \times |M_{e_{k+1}}j|)}}$$

where C_{ij} is the correlation coefficient between a node i in SOM M_{e_k} and a node j in SOM $M_{e_{k+1}}$ (note that i and j can represent the same node but in two different maps), X is the number of patterns that have moved from node i to j and $|M_{e_k}i|$ ($|M_{e_{k+1}}j|$) is the number patterns at node i (j) in SOM M_{e_k} ($M_{e_{k+1}}$). A migration is considered "interesting", and thus highlighted, if C is above a specified minimum relationship threshold (Min-Rel). With respect to the CTS application a threshold of 0.2 was found to provide a good working Min-Rel value. The Min-Rel value was also used to prune vertices and edges; any edge whose C value was below the Min-Rel value was not selected for depiction in the visualisation, similarly any vertex that has no links with a C value above Min-Rel was not depicted.

In the current extended implementation of Visuset nodes are depicted as: (i) single vertices (i.e. self links where the "migration" is from and to the same vertex), (ii) vertex pairs linked by an edge, (iii) chains of vertices linked by a sequence of edges, or (iv) more complex sub-graphs (islands). The size (diameter) of the vertices indicated the number of elements represented by that node in M_{e_k}. Some example output is presented in Fig. 4. This output features islands of SOM modes that are strongly connected by pattern migrations. Fig. 4 shows the migration of patterns from episode 2005 to episode 2006.

The proposed animation mechanism can be applied to pairs of visualisations (as described above) to illustrate the migration of patterns over three episodes (SOMs). At the start of an animation the display will be identical to the first visualisation (Map1) and will move to a configuration similar to the second visualisation (Map 2), although nodes will not necessarily be in the same display location. Thus the animations show how subsequent mappings change and consequently how the "pattern migration communities" change. As the animation progresses the correlation coefficients (C values) are linearly incremented or decremented from the values for the first map to that of the second map. Thus, as the animation progresses the links, nature of the islands, and overall number of nodes will change. For example if the correlation coefficient for a node in Map 1 is 0.3 and in Map 2 is 0.1 (assuming a threshold of 0.2) the node will "disappear" half way through the animation. Alternatively, if the correlation coefficient for a node in Map 1 is 0.1 and in Map 2 is 0.5 the node will "appear" a quarter of the way through the animation. Nodes that disappear and appear are highlighted in white and pink respectively.

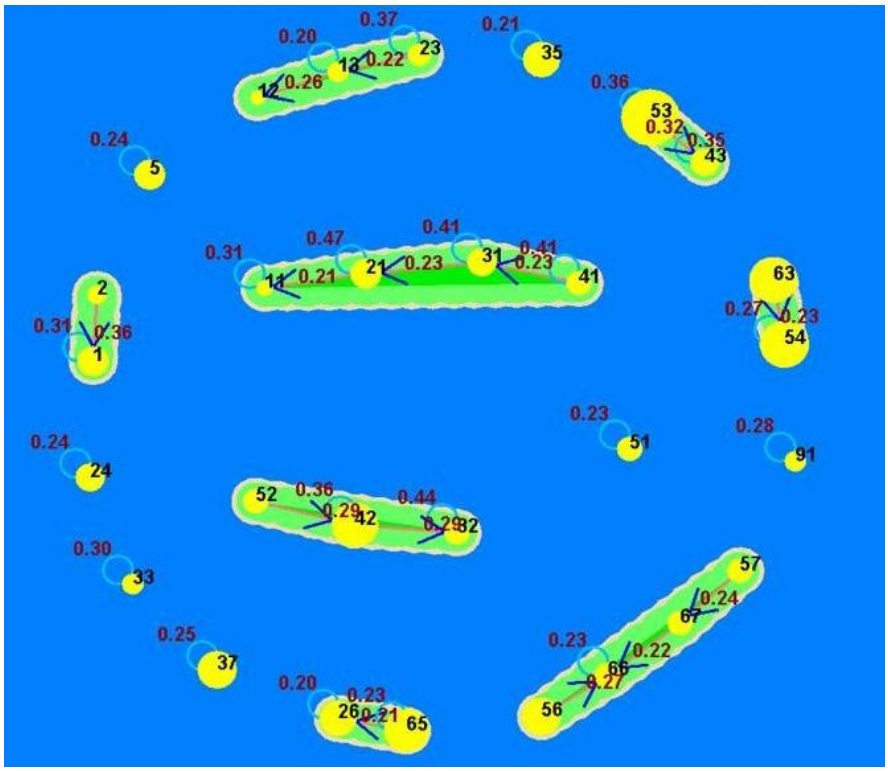

Fig. 4: *Visualisation of CTS pattern migrations from episode 2005 to episode 2006 in the CTS data*

10. Prediction Modelling

If we wish to model how information percolates across a network based on identified frequent patterns that exist in that network the patterns of interest must clearly include information about start and end locations and the nature of the traffic. We refer to these patterns as "combination patterns". Using TM-TFP, as already noted in Section 6, a great many trends are identified, only a subset of these are combination patterns; patterns that comprise location attributes associated with two different grid locations (vertices in the network) and movement attributes concerned with the flow of activity between these locations. Each relevant combination pattern will have an associated trend line defined in terms of a set of support values. The support counts are indicators of the frequency of traffic between the two nodes. This in turn may be interpreted as a probability measure indicating the likely traffic flow between the two locations.

Thus the prediction modelling process takes as input a set of trends associated with combination patterns. A set of "percolation matrices" are then generated. A percolation matrix comprises rows and columns representing the *fromLocation* and the *toLocation*, the row-column intersections then hold a numeric value indicating the probability of traffic flow between the indicated nodes. A percolation matrix thus comprises L × L elements, where (at least in the context of the CTS application) L is the number of possible location grid IDs. The magnitude of L is thus dependent on the number of possible locations represented in the set FP. A set of n percolation matrices (one per time stamp) is generated.

```
Input: FP = Set of Frequent combination patterns, set of Trends
Output: n Percolation Matrices generated from FP

1) for ∀ fp ∈ FP do
2)        Extract probability (p) of each fp from its associated trend;
3) end
4) for k←1 to |Trends| do
5)         Construct a matrix of size N × N );
6)         for i ← 1 to | F P | do
```

```
7)              Insert pi into the matrix k at the appropriate location;
8)      end
9)  end
```

Algorithm 8: *Percolation matrix construction*

Algorithm 8 describes the process of building the percolation matrix and extracting the probability of movement so as to facilitate the desired Prediction Modelling (PM). The algorithm first extracts the probability of each pattern fp in FP (Line 2). Recall that the support values associated with each time stamp n defines the probability of traffic flowing between nodes (a support value of 0 is used to indicate a value below the threshold σ). Thus all support values for the given frequent patterns are interpreted as probability values (p). Therefore, given a specific frequent pattern fp_i, the probability of traffic between the two reference locations p_i is defined as:

$$p_i = \frac{\sup port(fp_i)}{\sum fp_i}$$

Thus, $p_1 + p_2 + \ldots + p_n = 1$. Once the probabilities for all fp have been identified, the algorithm constructs the percolation matrix (line 3) and the probability values inserted into the matrix. The process repeats until all n percolation matrices are constructed. The percolation matrices are then used as the input to the Visualisation module described in the next section.

11. Prediction Visualisation

Once the percolation matrix, as described above, has been generated the predictions can be visualized. The PTMF features two types of visualisation: (i) "probability maps" generated using a further extension of the Visuset software system, and (ii) geographical maps using Google Earth. The aim of the visualisation is to demonstrate, in a clear and straight forward manner, how information travels across a given network. The further extension of the Visuset system provided an interpretation of the probability matrix data in the form of a "probability map" that illustrates a given node and link structure. The map highlights which nodes are connected directly (and, by extension, indirectly) to other nodes using "weighted" links. The weightings are determined from the probabilities contained in the generated percolation matrices. The configuration of Visuset used for the purpose of prediction modelling was similar to the configuration described above, however, in this case the numbers of patterns in each node were ignored as the significant information to be displayed are location nodes, traffic links and the identified probability values.

The version of Visuset described above includes a mechanism to identify communities of nodes in the network. However, using the percolation matrix it is straightforward to identify nodes that are connected together as this information can be extracted directly from the matrix. Thus, the probability maps illustrate groups of nodes that are connected together which, in the same manner as described above, are depicted as "islands". An example probability map is given in Fig. 5 for the month of October 2006. The islands indicate strongly connected nodes. Thus in a bovine disease monitoring situation we can expect the disease to spread rapidly across "islands" and less rapidly with respect to unconnected nodes.

From the generated maps, the following can be identified:

1) **One step percolation**: Paths describing how information or events may travel between nodes (in one "step"). In the proposed PM, a one step percolation is defined as a direct link between a pair of nodes, a and b.
2) **Probability values**: Probability values describing the likelihood that information may travel between a particular pair of nodes. The probability values can also be used to calculate the probability of information flows encompassing two or more steps. A two step percolation refers to movement between two pairs of connected nodes. We may also be able to identify three and four step percolations (n-step percolations). These types of percolation are collectively

described as “complex” connections.

3) **Node communities**: Communities of nodes that are connected together.

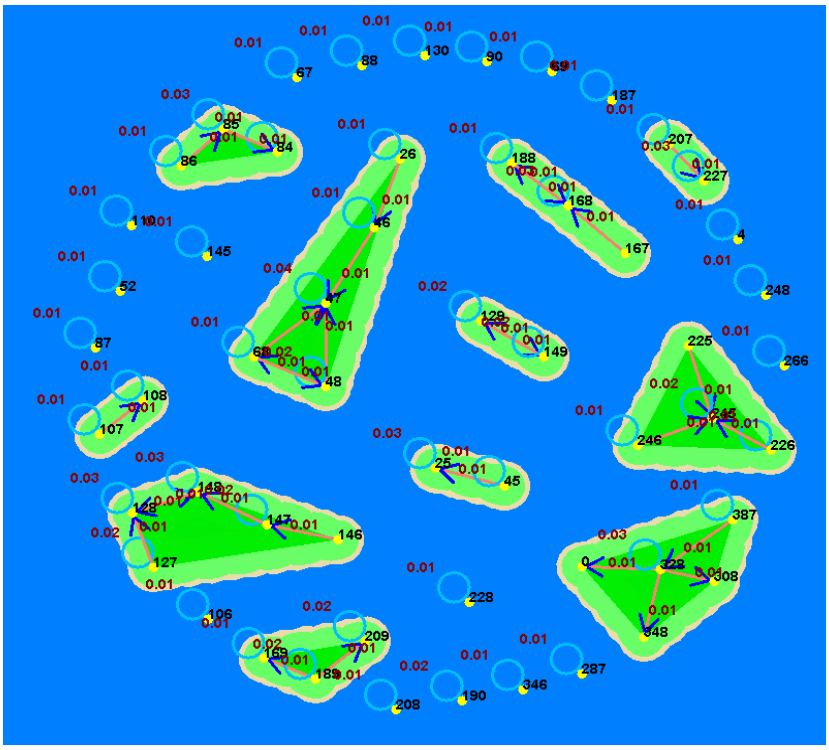

Fig. 5: *Combination Patterns Probability Map for the CTS data for October 2006*

12. Summary and Conclusions

In this paper, the PTMF has been described, a framework for identifying trends and making predictions with respect to social network data. The main contributions of the work may be itemized as follows:

1) A mechanism for generating temporal spatial frequent patterns and trends that may exist within a network, in terms of episodes.
2) A mechanism for clustering groups of trends, using a SOM technique, so as to assist in the further analysis of the identified trends.
3) A trend cluster analysis mechanism to support the detection of changes in trends and frequent pattern migrations.
4) A visualization of pattern movement (traffic) from one trend cluster to another over a period of time, again to facilitate and support trend analysis.
5) A prediction modelling and visualisation technique that can be applied to network data, which illustrates the manner in which information (events) might travel across a (social) network.

The work described has been illustrated with respect to the GB cattle industry, and more specifically the GB Cattle Tracking System (CTS) database; however, the work has clear application with respect to other industries. Although not reported in this paper the authors have also conducted experiments using the customer base of an insurance agency and a military logistics (supply) network; with similar results.

In the future work, the authors would like to optimise the design of experiments with a proper implementation such as that presented in [39]. As mentioned in Section 7, there is no scientific method for determining the best values for n and m. However, there are some methods that the authors would like to investigate; one of them is the "Elbow-criterion", which is use to compare the Sum of Squared Differences (SSD) for different optimum number of clusters in clustering data.

References

[1] P. Domingos. Mining social networks for viral marketing. IEEE Intelligent Systems, 20(1):80-82, 2005.

[2] M.A. Russell. Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites. O'Reilly Media, 2011.

[3] H. Becker, D. Iter, M. Naaman, L. Gravano: Identifying content for planned events across social media sites. In Proceedings of the fifth ACM international conference on Web search and data mining (pp. 533-542). ACM, 2012.

[4] A. McCallum, X. Wang, and A. Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. Journal of Artificial Intelligence Research, 30(1):249-272, October 2007.

[5] I. Ting, T. Hong and L.S.L. Wang. Social network mining, analysis, and research trends: techniques and applications. Publisher: IGI Global, ISBN: 978-1613505137, 2011.

[6] P. Nohuddin, R. Christley, F. Coenen, Y. Patel, C. Setzkorn, and S. Williams. Finding "interesting" trends in social networks using frequent pattern mining and self-organizing maps. Journal of Knowledge Based Systems: Special Issue, 29:104-113, 2011.

[7] P. Nohuddin, F. Coenen, R. Christley, W. Sunayama: Visualisation of Trend Pattern Migrations in Social Networks. Advances in Visual Informatics, 77-88, 2015.

[8] H. Habiba, Y. Yu, T. Berger-Wolf, and J. Saia. Finding spread blockers in dynamic networks. In Proceedings of the Second international conference on Advances in social network mining and analysis, volume 08, pages 55–76. Springer, 2008.

[9] A. Z. Aliabadi, F. Razzaghi, S. P. Madani K., A. A. Ghorbani: Classifying Organizational Roles Using Email Social Networks. Advances in Artificial Intelligence. Vol. 7884 pp 301-307, 2013.

[10] P. Symeonidis, E. Tiakas, Y. Manolopoulos: Product recommendation and rating prediction based on multi-modal social networks. In *Proceedings of the fifth ACM conference on Recommender systems* (RecSys '11). ACM, New York, NY, USA, 61-68, 2011.

[11] Y. Liang, J. Caverlee, Cheng Cao: A Noise-Filtering Approach for Spatio-temporal Event Detection in Social Media. Advances in Information Retrieval, Volume 9022 pp 233-244, 2015.

[12] M. Khan, F. Coenen, D. Reid, H. Tawfik, R. Patel, and A. Lawson. A sliding windows based dual support framework for discovering emerging trends from temporal data. Journal of Knowledge Based System, 23(4):316-322, 2011.

[13] V. Somaraki, D. Broadbent, F. Coenen, and S. Harding. Finding temporal patterns in noisy longitudinal data: A study in diabetic retinopathy. In Proceedings 10th Industrial Conference on Data Mining, pages 418-431, 2010.

[14] A. Gosain and A. Kumar: Analysis of health care data using different data mining techniques, Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. Chennai, 2009, pp. 1-6. 2009.

[15] S.A. Nunes, L.A.S. Romani, A.M.H. Avila, C. Traina Jr, E.P.M. de Sousa: Fractal-based analysis to identify trend changes in multiple climate time series, Journal of Information and Data Management 2 (1), 51, 2011.

[16] M. Angermeyer and H. Matschinger: Causal beliefs and attitudes to people with schizophrenia: Trend analysis based on data from two population surveys in Germany. The British Journal of Psychiatry, 186:331–334, 2005.

[17] P. Gloor, J. Krauss, S. Nann, K. Fischbach, and D. Schoder. Web science 2.0: Identifying trends through semantic social network analysis. social science re- search network. Social Science

Research Network Working Paper Series, 4:215– 222, 2008.

[18] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behavior of recommender systems. Journal of Knowledge Based System, 23(6):520-528, 2010.

[19] W. Yuan, D. Guan, Y. Lee, S. Lee, and S. J. Hur. Improved trust aware recommender system using small worldness of trust networks. Journal of Knowledge Based System, 23:232-238, April 2010.

[20] C. Kaiser, S. Schlick, and F. Bodendorf. Warning system for online market research - identifying critical situations in online opinion formation. Knowledge Based System, 24:824-836, August 2011.

[21] J. Han, M. Kamber, and J. Pei. Data Mining: Concepts and Techniques 3rd Edition. Morgan Kaufmann, 2011.

[22] J. Neville and F. Provost. Prediction modelling in social networks. ICWSM 2009 Tutorial, 2009.

[23] B. Chen, Q. Zhao, B. Sun, and P. Mitra. Predicting blogging behavior using tem- poral and social networks. In Proceedings of 2007 IEEE International Conference on Data Mining, pages 439–444, 2007.

[24] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In Neural Information Processing Systems, 2003.

[25] V. Lampos and N. Cristianini. Tracking the flu pandemic by monitoring the social web. In 2nd IAPR Workshop on Cognitive Information Processing, pages 411-416. IEEE Press, June 2010.

[26] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: Improving geo- graphical prediction with social and spatial proximity. In Proceedings of the 19th International Conference on World Wide Web, pages 61-70, 2010.

[27] R. Agrawal, T. Imielinski, and A. Swami: Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216. ACM Press, 1993.

[28] R. Agrawal and R. Srikant: Fast algorithms for mining association rules. In Proceedings of 20th International Conference on Very Large Data Bases, pages 487-499, 1994.

[29] F. Coenen, G. Goulbourne, and P. Leng. Computing association rules using partial totals. In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, pages 54-66, 2001.

[30] F. Coenen, P. Leng, and S. Ahmed. Data structures for association rule mining: T-trees and p-trees. IEEE Transactions on Data and Knowledge Engineering, 16(6):774-778, 2004.

[31] T. Kohonen. The self organizing maps. In Springer Series in Information Science, volume Vol. 30, 1995.

[32] T. Kohonen. The self organizing maps. Neurocomputing Elsevier Science, 21:1-6, 1998.

[33] M. Cottrell and P Rousset. A powerful tool for analyzing and representing multi- dimensional quantitative and qualitative data. In Proceedings of the International Work-Conference on Artificial and Natural Neural Networks: Biological and Ar- tificial Computation: From Neuroscience to Technology, pages 861-871, 1997.

[34] A. Chin and M. Chignell. Identifying communities in blogs: roles for social network analysis and survey instruments. International Journal of Web Based Communities, 3(3):345-363. 2007.

[35] C. Tantipathananandh, T. Berger-Wolf and D. Kempe. A framework for community identification in dynamic social networks. Proceedings 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07), pages 717-726, 2007.

[36] M. Newman. Fast algorithm for detecting community structure in networks. Physical Review

E, 69:1-5, Jun 2004.

[37] T. Nishikido, Sunayama W. and Y. Nishihara. Valuable change detection in keyword map animation. Proceedings 22nd Canadian Conference on Artificial Intelligence, pages 233–236. Springer-Verlag, 2009.

[38] K. Sugiyama and K. Misue. Graph drawing by the magnetic spring model. Journal of Visual Languages and Computing, 6(3):217-231, 1995.

[39] A. Ochoa-Zezzatti, J. Sánchez, A. Hernández-Aguilar, R. Pérez. Improving an Industrial problem optimizing the material in car seats. International Journal of Combinatorial Optimization Problems and Informatics, vol. 7 (1), 54-62, 2016.