Discovering Temporal Patterns from Insurance Interaction Data

Maleeha Qazi,[2] Srinivas Tunuguntla,[1] Peng Lee,[2] Teja Kanchinadam,[2]
Glenn Fung,[2] Neeraj Arora[1]
[1]University of Wisconsin - Madison
[2]American Family Insurance
{stunuguntla,neeraj.arora}@wisc.edu, {mqazi,plee,tkanchin,gfung}@amfam.com

Abstract

In the insurance industry, timely and effective interaction with customers are at the core of everyday operations and processes that are key for a satisfactory customer experience. These interactions often result in sequences of data derived from events that occur over time. Such recurrent patterns can provide valuable information that can be used in a variety of ways to improve customer related work-flows. In this paper we demonstrate the application of a recently proposed algorithm to uncover such time patterns that takes into account the time between events to form such patterns. We use temporal customer data generated from two different use-cases (satisfaction and fraud) to show that this algorithm successfully detects patterns that occur in the insurance context.

1 Motivation/Introduction

Recurrent patterns in event sequences occur in diverse contexts that include notes in a musical composition, moves such as give-and-go in sports and turn-taking behavior exhibited by humans. In a retail context, buyers may engage in predictable on-line browsing behavior patterns before making a purchase. Similarly, in many processes and work-flows in the insurance industry, policyholders of an insurance provider may experience distinct patterns of interactions as they navigate the claims process. Examples of events during this process include claim reporting, bill paying, adding a coverage, and general inquires. Patterns extracted from the temporal data in the claims process reveal insights that could be used to optimize customer interactions and enhance the overall customer experience with the insurance company. The purpose of this paper is two fold:

1. To introduce the recently proposed scalable T-pattern algorithm (Arora, Fung, and Tunuguntla 2017) that has been proposed in a business setting to the AI and data mining community and to relate it to other similar methods for temporal discovery.
2. To showcase how this proposed algorithm can be used to extract patterns from temporal sequence data that arise in the insurance domain.

The recently proposed algorithm (Arora, Fung, and Tunuguntla 2017) builds upon prior work (Magnusson 2000) which defined the term *T-pattern* as a sequence of events that occur with a relatively consistent time interval between each event and where the sequence appears in the data more often than one would statistically expect by chance alone. The time dimension adds significant complexity to the task of detecting patterns in any data, and identification of such patterns becomes quite difficult for even modest sized data. Most algorithms known in the data mining community either assume that the time intervals between events is constant (Mannila, Toivonen, and Inkeri Verkamo 1997), or ignore the time intervals between events and focus only in the sequential order of the events (Zaki 2000; 2001).

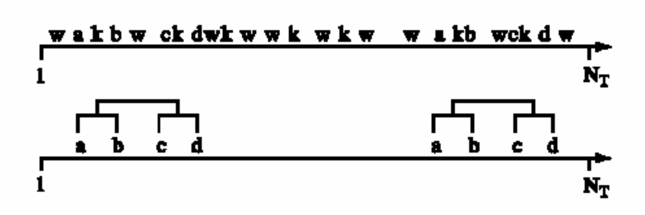

Figure 1: Illustration from (Magnusson 2000)

A simple illustration from (Magnusson 2000) helps motivate the thinking behind a *T-pattern*. In this illustration, there are six events (a, b, c, d, w, k) that occur several different times during the period $[1, N_T]$ as shown in the first row in Figure 1. The second row reveals the time pattern among some of the variables. At the first level of the hierarchy, two T-patterns are found: $[a, b]$ and $[c, d]$. The second level of the hierarchy reveals one more T-pattern: $[a, b]$, $[c, d]$. Even in this simple example, the underlying T-pattern is not self-evident from the observed data. As we will show, under this paradigm, the problem of detecting a T-pattern gets exponentially difficult as the number of events in the data increase.

The rest of the paper is laid out as follows: In section 2, we summarize relevant related prior work and in section 3, we outline the original T-pattern algorithm (Magnusson 2000). We identify several limitations that preclude this algorithm from being usable for typical business problems and offer possible solutions. In section 4 we summarize our algorithm that proposes a solution for each limitation. This algorithm is scalable to typical business problems that have a large number of events and individuals.

In (Arora, Fung, and Tunuguntla 2017) we conducted simulations to exhibit the properties of the proposed algorithm and it's ability to recover T-patterns when the true T-patterns are known. In the empirical section of this paper, section 6, we test the algorithm using customer insurance data derived form two different use-cases. The final section of the paper summarizes the main aspects of the paper and offers conclusions.

2 Related Work

Over the years the T-pattern algorithm has been applied in a variety of contexts to study interactions and behavior involving animals and humans. (Casarrubea et al. 2015) does an excellent job of summarizing these applications. The research involving T-patterns and humans span topics such as cognitive and social disorders, gender relations, team effectiveness, and sports. In their paper on T-patterns involving performance of soccer teams, (Borrie, Jonsson, and Magnusson 2002) observe that traditional analyses focus on frequency of event occurrence (e.g. number of passes, assists, unforced errors, etc.) as the index of performance. What's missing from such analyses, they note, is the temporal structure and interrelationships between events. Using the T-pattern approach, the authors point to the importance of temporal aspects of sports performance that reveal novel patterns that impact team performance. We believe that such temporal patterns are widely prevalent in business applications and are important to study.

In (Batal et al. 2016), the authors present Recent Temporal Pattern (RTP) mining as a "novel approach for efficiently finding predictive patterns for event detection problems" in complex multivariate temporal data like electronic health records (EHRs). Although their methodology has some similarities to our approach, for example using supervised pruning to help scalability, easy interpretability of detected patterns, and bottom-up detection of patterns, their technique is built on the assumption that more recent patterns are more predictive than past patterns. This assumption holds for the medical domain but doesn't necessarily apply to domains such as insurance. Also, their algorithm requires abstracting the concept of time in two different ways: trend abstractions and value abstractions. In addition, the concept of "recent" is declared using a user-defined parameter. Our algorithm, in contrast, is not limiting in this aspect. Time doesn't require a user-defined restriction or abstraction in our approach and the purpose of our algorithm is to find the most salient time boundaries to define the detected patterns.

Various algorithms known in the data mining community have been compared to one another in previous surveys (Roddick and Spiliopoulou 2002; Mooney and Roddick 2013). In Table 1 we chose three well known algorithms, WINEPI/MINEPI (Mannila, Toivonen, and Inkeri Verkamo 1997), SPADE/cSPADE (Zaki 2000; 2001) & RTP (Batal et al. 2016), to compare against our algorithm along some key dimensions of interest. As can be seen, WINEPI/MINEPI & SPADE/cSPADE do not represent time explicitly in their output rules. Although RTP does represent time in its output rules, it is re-represented as pre-defined relationships (e.g. co-occurs, contains, etc.) This is a common trend for time series data. It can limit the knowledge the rules can represent to what a human has given an algorithm access to represent. If knowledge isn't represented at the correct granularity for learning, then the algorithm cannot produce the desired results. This is one of the reasons we allow time to remain as-is within our algorithm.

As far as we know, our proposed algorithm is the only algorithm (other than Magnusson's) that considers both order and variable time between events, without having to re-represent time in any way, & while being able to scale to address the demand of many present-day business problems.

3 Original T-pattern Algorithm

We begin this section by formalizing the definitions used by our proposed algorithm (Arora, Fung, and Tunuguntla 2017) and the algorithm proposed by (Magnusson 2000). Let event i occur for person j during time t. We define the following variable to capture the occurrence of event type E.

$$E_{ijt} = \begin{cases} 1 & \text{if event } i \text{ for person } j \text{ occurs during time } t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

As an example, for four individuals (P_1 to P_4) and nine event types (E_1 to E_9). For $i = 1, ..., 9$ events, the possible set of T-patterns is Q and includes all possible event combinations.

$$Q = \{(E_1, E_2), (E_1, E_3), .., (E_8, E_9), .., (E_1, E_2, .., E_9), \\ .., (E_2, E_1), .., (E_9, .., E_3, E_2, E_1)\} \tag{2}$$

The goal is to detect T-patterns such as (E_1, E_2) that follow the definition that a T-pattern is (i) a sequence of events that occur with a relatively consistent time interval between each event and (ii) where the sequence appears in the data more often than one would statistically expect by chance alone. We use the subscript k to refer to a T-pattern where $k \in \{1, ..., K\}$ is a small subset of all possible event combinations in Q.

The T-patterns are not expected to follow a deterministic time pattern. That is, the inter-event time maybe stochastic. One of our goals is to find a *critical interval* that corresponds to each T-pattern. Formally, for a two-event T-pattern k we will uncover the interval (c_{k1}, c_{k2}) such that the second element of the T-pattern falls somewhere between $t = c_{k1}$ and $t = c_{k2}$ after the first element. For example, $E_1(4, 10)E_2$ implies that E_2 is likely to occur 4 to 10 time units after E_1. The same notations naturally extends to longer T-patterns that exceed two events. For example, $E_3(3, 11)E_5(2, 5)E_7$ implies that E_5 is likely to occur 3 to 11 time units after E_3 and E_7 is likely to occur 2 to 5 time units after E_5.

3.1 Magnusson's Algorithm

For any pair of events E_i and $E_{i'}$, next we outline how the algorithm originally proposed by (Magnusson 2000) proceeds to determine whether the two events follow a T-pattern and if so, what is the corresponding critical interval. The test for a

Table 1: Comparison of Various Algorithms

Comparison Factors	**ŴINEPI**	**MINEPI**	**SPADE/cSPADE**	**RTP**	**T-pattern**
Algorithm family	Temporal Sequence	Temporal Sequence	Apriori based	Temporal Sequence	Temporal Sequence
Association Discovery Mode	Inter-transaction	Inter-transaction	Intra-transaction	Inter-transaction	Inter-transaction
Gap allowed between individual elements in sequence	Syntax gives the maximum total rule time, including both antecedent and consequent; no information about the time from observation of the antecedent to the consequent	Syntax gives the ability to find different rules based on the time between the occurrences in the antecedent, e.g. if the events A and B occur within 10 minutes this results in C, but if they occur within 45 minutes it will most likely result in Y	max/min gap between each set of elements kept the same	max/min gap between each set of elements kept the same	max/min gap between each set of elements allowed to be distinct
Order of events matters	Yes	Yes	Yes	Yes	Yes
Time representation	Used as input, but not represented in rules output	Used as input, but not represented in rules output	Used as input, but not represented in rules output	Time represented as pre-defined relationships between elements (e.g. co-occurs)	Used as input, and represented numerically in output
Example Rule	A, B $\rightarrow$ C (min: 10, avg: 30, max: 60) (sup: 7%, conf: 90%, lift: 7)	A, B $\rightarrow$ C (min: 10, avg: 30, max: 60) (sup: 7%, conf: 90%, lift: 7)	{A,B},{C}	A=H co-occurs B=H $\rightarrow$ C=RENAL	{A(1,4)B}(3,8)C

T-pattern is based on the null hypothesis that the two events E_i and $E_{i'}$ are distributed independently with the observed frequency during the observation period $[1, T]$. The following probabilities are necessary to test for the presence of T-pattern $(E_i, E_{i'})$. The marginal probabilities for $E_{i'}$ are given by:

$$Pr(E_{i'} = 1) = \frac{N_{E_{i'}}}{N_T} \; ; Pr(E_{i'} = 0) = 1 - \frac{N_{E_{i'}}}{N_T} \quad (3)$$

For the event E_i, let us focus our attention on an arbitrary interval $[c_1, c_2]$ that follows E_i. The probability that one or more events $E_{i'}$ lie in the interval $[c_1, c_2]$ that follows E_i, is calculated as follows,

$$Pr(E_{i'} \geq 1 | c_1, c_2) = 1 - (1 - \frac{N_{E_{i'}}}{N_T})^{c_2 - c_1 + 1} \quad (4)$$

Let $n_{E_i,E_{i'}}$ be the number of instances when $E_{i'}$ occurs within the interval $[c_1, c_2]$ after E_i. Using equation 4, one can calculate the probability distribution of $n_{E_i,E_{i'}}$.

$$\begin{aligned} Pr(n_{E_i,E_{i'}} = n | c_1, c_2) = Binomial(N_{E_i}, n, \\ Pr(E_{i'} = 1 | c_1, c_2)) \end{aligned} \quad (5)$$

For the N_{E_i} occasions that event E_i occurs, the expected number of times the event $E_{i'}$ occurs within the interval $[c_1, c_2]$ after E_i can be written as

$$E(n_{E_i,E_{i'}}) = N_{E_i} \times (1 - (1 - \frac{N_{E_{i'}}}{N_T})^{c_2 - c_1 + 1}) \quad (6)$$

Given that we observe the number of times $E_{i'}$ occurs within the interval $[c_1, c_2]$, the probability that the $(E_i, E_{i'})$ is *not* a T-pattern is given below.

$$p = 1 - \sum_{n=0}^{N_{E_i,E_{i'}} - 1} Binomial(N_{E_i}, n, Pr(E_{i'} = 1 | c_1, c_2)) \quad (7)$$

This is also the probability of the null hypothesis that the two events E_i and $E_{i'}$ are distributed independently to be true. Therefore, small values of p in Magnusson's algorithm indicate evidence in support of a T-pattern.

The algorithm begins by recording the time until the nearest occurrence of $E_{i'}$ after E_i and defines the distribution of these distance intervals. In doing so, it ignores any event that occurs between E_i and $E_{i'}$. Then, the algorithm checks all the possible time intervals $(c_k, c_{k'})$ where $E_{i'}$ occurs after E_i for statistical significance. If an interval is found to be significant based upon the test criterion explained in Equation 7, the candidate pattern $E_i(c_l, c_{l*})E_{i'}$ is called a T-pattern and saved as a candidate to be part of a more complex t-pattern in future iterations.

3.2 Limitations of Magnusson's Algorithm

The original algorithm suffers from a series of limitations when applied to typical business contexts, and we outline them here:

1. **Scalability**: The algorithm is usually applied to a few individuals with a limited number of events and is ill-suited for data that are larger in scope - including many individuals and events. As the number of events and individuals increase, the algorithm breaks down because the number of combinations increase exponentially.
2. **Heterogeneous individuals**: The original algorithm fails to recognize that events occur at different frequencies for different individuals. As a result, the algorithm uncovers T-patterns that exhibit long critical intervals that do not occur very frequently.
3. **Distributional assumptions**: The test for a pattern E_i, $E_{i'}$ is based on the null hypothesis that the two events are independent Bernoulli processes over the observation period with constant probabilities of occurrence within a time unit. The original algorithm (see Equation 3) uses

this to calculate the probability of an event $E_{i'}$ occurring in the interval $[d1, d2]$ after an event E_i. We note that, since the nearest occurrences of $E_{i'}$ after E_i are used to calculate the distance intervals, under the null hypothesis, these distance intervals should be geometrically distributed. We incorporate this correction from the Bernoulli to a geometric distribution in the algorithm.

4 A Revised T-pattern Algorithm

Our revised T-pattern algorithm is documented in (Arora, Fung, and Tunuguntla 2017) and offers several improvements to address the limitations pointed out above. These improvements are summarized below.

4.1 Scalability

Our proposed algorithm addresses this problem by: (a) optionally incorporating the use of labels in a supervised manner to discard patterns that have poor or no discriminative value for classification problems, (b) proposing an adequate data structure that makes calculations more tractable, and (c) parallelizing the algorithm. In this section we will focus on (a) while (b) and (c) will be addressed in section 5.

A challenging aspect of typical business applications is that the number of individuals and events maybe large. For example, the data we use in our experimental section involves thousands of samples (individuals, observations), hundreds of event types, and millions of total event occurrences. This imposes substantial computation burden on the algorithm. The challenge is presented by the large set size for Q (the total number of T-pattern candidates to evaluate) and critical interval determination for each possible T-pattern. Our proposed solution to tackle this combinatorial hurdle is to use "smart" pruning. For our revised T-pattern algorithm, both unsupervised and supervised pruning proves valuable in scaling and hence avoiding, or reducing, the possibility of a combinatorial explosion that may lead to intractability.

When applying **unsupervised pruning** the idea is pretty simple: T-patterns that may have very low event counts or are associated with events with significant low counts may offer little or no information. In contrast, their inclusion may result in substantial computational cost. For industrial applications, an important application for T-patterns is in the area of classification problems.

When applying **supervised pruning**, we are interested in measuring the information gain from the occurrence of a candidate pattern $(E_i, E_{i'})$ to a given label by using the Kullback-Leibler (KL) divergence criterion (Kullback and Leibler 1951). This allows us to uncover T-patterns that are likely candidates to be the drivers of the difference between two classes. Such a supervised pruning approach is general and could be applied to any classification target variable or label (e.g. satisfied/unsatisfied customers, loyal/switchers, etc.).

4.2 Heterogeneous Individuals

To account for the fact that the number of events per individual vary, the probability of an event $E_{i'}$ occurring in the interval $[c_1, c_2]$ after E_i is calculated at the individual level.

Unlike the binomial distribution that involves a sequence of Bernoulli trials with the same success probability for each individual, the Poisson Binomial distribution involves a sequence of Bernoulli trials with a different success probability for each individual. So we chose to replace the use of the Bernoulli distribution with the Poisson Binomial distribution.

Hence, the evidence against the T-pattern is given by

$$p = 1 - \sum_{n=0}^{N_{E_i,E_{i'}}-1} PoissonBinomial(N_{E_i}, n, Pr) \quad (8)$$

Small values of p indicate evidence in support of a T-pattern.

4.3 Distributional Assumptions

Starting at an arbitrary point in time, the distance of the first occurrence of event E_i after that point, follows a geometric distribution with success probability P_{E_i}. Similarly, the distance to the first occurrence of event $E_{i'}$ after E_i, follows a geometric distribution with parameter P_{E_i}. An event $E_{i'}$ is paired with an event E_i, at a distance d only if it is the first occurrence of $E_{i'}$ after E_i. It can be shown (Arora, Fung, and Tunuguntla 2017) that the probability of an event $E_{i'}$ occurring in the interval $[c_1, c_2]$ after an event E_i can be calculated as

$$\begin{aligned} Pr(c_1 \leq D \leq c_2) &= Pr(D \leq c_2) - Pr(D \leq c_1 - 1) \\ &= (1 - p_{dist})^{c_1 - 1} - (1 - p_{dist})^{c_2} \end{aligned} \quad (9)$$

We therefore replace Equation 4 by Equation 9 in the original algorithm.

4.4 CI Search Stopping Criteria

In the critical interval (CI) search algorithm, (Magnusson 2000) suggests that the search process should stop when a critical interval is found to be significant at an alpha level. We find this stopping criteria to be somewhat arbitrary and associated with a cost. Since the search process starts with the longest possible interval and iteratively reduces it until a significant interval is found, we find that the resulting T-patterns often exhibit long critical intervals. A T-pattern with a shorter critical interval, that also occurs more frequently, often goes undetected. To overcome this problem we propose that the search process should continue to find the interval that has the *lowest* p-value.

4.5 Summary of Proposed Algorithm

We start by treating each event type as a 1-event T-Pattern. For each event type, the start time and end time for each of its instances are recorded for all the individuals in the dataset. In general, a candidate pattern is of the form (P_i, P_j) where P_i, P_j are two T-Patterns found in the data. The algorithm can be summarized in the following steps:

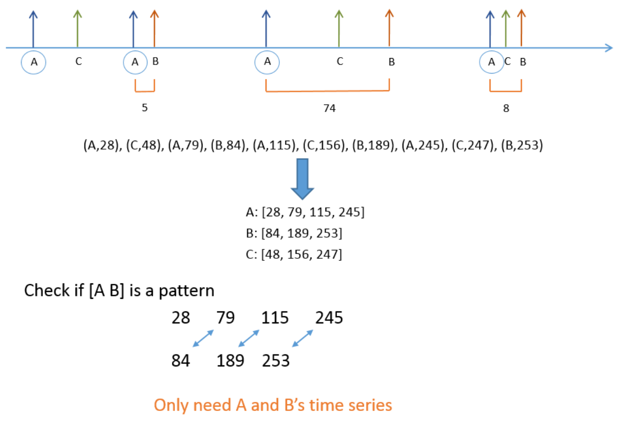
A
C
A
B
A
C
B
A
C
B
5
74
8
(A,28), (C,48), (A,79), (B,84), (A,115), (C,156), (B,189), (A,245), (C,247), (B,253)
A: [28, 79, 115, 245]
B: [84, 189, 253]
C: [48, 156, 247]
Check if [A B] is a pattern
28 79 115 245
84 189 253
Only need A and B's time series

Figure 2: Data Restructuring: Start with events stored in the increasing order of their times of occurrence, then sample pairs to see if pattern exists.

Algorithm 1 Revised Scalable T-pattern Algorithm

```
Input: Vector of targets or labels $L$ - $\{l_1, l_2, \ldots, l_m\}$
       Matrix of events $M$
Output: Final list of T-patterns
 1: Create a list of candidate patterns $C$.
 2: for each candidate pattern $(P_i, P_j) \in C$ do
 3:     Create a list of event pair instances for all the individuals. The event pair instances are of the form $(P_i(d)P_j)$, where $d$ is the distance between the event instances $P_i$, $P_j$.
 4:     Evaluate if the candidate pattern is a T-Pattern.
 5:     if Candidate is a t-pattern then
 6:         calculate the critical interval $(d_1, d_2)$ and add the pattern $(P_i, P_j)$ to the list of found T-Patterns $T$.
 7:     end if
 8:     add the top-$k$ features to the BN in a Naive Bayes fashion.
 9: end for
10: Go to step (1) if any new T-Patterns are found in step (2)
11: return $T$
```

5 A Parallel and Scalable Implementation

5.1 Data Structures

In algorithm 1, step (3), for a given candidate pattern (P_i, P_j), we create a list of all event pair instances of the form $(P_i(d)P_j)$, where d is the distance between the end time of an instance of P_i and the start time of the instance of P_j that immediately follows. To improve the computational efficiency of this step, we use the following data structure to store data corresponding to any pattern (see Figure 2). The instances of the pattern are stored in the increasing order of their times of occurrence. The order is the same irrespective of using the start times or the end times of the instances to sort, because the instances are non-overlapping. For each instance, the start and end times are recorded.

For a given candidate pattern (P_i, P_j), to calculate the event pair instances as in algorithm 1, step (3), it is sufficient to perform a simultaneous linear search on the instances of both the patterns, because the instances are sorted. The event pair instances now form the instances of the candidate pattern (P_i, P_j). The start time of an instance of the form $(P_i(d)P_j)$ is the start time of the instance of P_i. Similarly, the end time of the instance is the end time of the instance of P_j. If the candidate pattern is found to be a T-Pattern with critical interval (d_1, d_2), the event pair instances that satisfy $d_1 \leq d \leq d_2$ are collected, sorted and stored as data for the new T-Pattern.

5.2 Parallel Implementation

With large datasets that include many event types, the algorithm requires significant computation because of the need to evaluate many candidate patterns. With the ease of access to cloud computing and the availability of multiple cores on each computer, parallelization can be used to speed up the algorithm significantly.

In algorithm 1, step (2), we evaluate whether each of the candidate patterns is a T-Pattern. The evaluation of a candidate pattern is independent of the evaluation of other candidate patterns. Hence, this step can be easily parallelized. Each candidate pattern can be processed on a different core of a computer. In the implementation, a master thread and a number of worker threads are used to evaluate candidate patterns efficiently. The master thread creates a queue of candidate patterns. This is a FIFO queue and is serviced by the worker threads. The worker threads watch the queue and pick up a candidate pattern to evaluate whenever they become available. If a candidate pattern is found to be a T-Pattern, it is returned to the master thread. All the T-Patterns that are found in Algorithm 1, step (2) are collected by the master thread and the process is repeated. We use a cluster of 16 computers that use Message Passing Interface (MPI) protocol to communicate. MPI is a communication protocol that supports both point-to-point and collective communication in parallel programming. Each computer has 16 cores and has its own private memory of 60 GB. We choose the MPI protocol over the others because it allows the computers in the cluster to read data in parallel and store it in their private memory. The use of the cluster reduced the run time of the algorithm from approximately 16 days on a single computer to less than 2 hours.

6 T-patterns from Insurance Work-flows

Next, we use temporal customer data generated from two different insurance-related use-cases to show that the T-pattern algorithm successfully detects discriminative patterns that occur in the context of insurance work-flows.

6.1 Customer Satisfaction

We selected claims data from a well-known insurance company to test the T-pattern algorithm. In order for an insurance policy holder to get reimbursed for a loss or an incident, a claim has to be filed. The claim process usually consists of a series of events and interactions that involve the insurance company, the insured, and various third parties. The process ends shortly after a payment is adjudicated.

According to Accenture in their news release of Oct. 13, 2014: "83 percent of dissatisfied customers [after a claim] are planning to switch or have already switched to another insurer". Given that the insurance company mentioned had 142K property and 644K auto claims submitted in 2014, even with low levels of dissatisfaction, capturing these early on can result in fewer households leaving the company after a claim. This in turn can translate to roughly 4 million dollars in business a year.

We use the concept of a *journey-map* as a data-driven structured time-line where all the events pertinent to the claim process are identified and positioned temporally with respect to each other. To generate the dataset used for this problem we created claim journey-maps. These journey-maps integrate events from different data sources that are available during the claim process. Most of these events are internal to the company and some involve the customer or a third party (e.g. a car repair shop). We have two primary purposes in these analyses. First, we want to show that the proposed algorithm can detect T-patterns that routinely occur in the context of insurance claims. This would serve as a face validity check for the algorithm. Second, we want to uncover T-patterns that separate dissatisfied customers from satisfied ones. Such T-patterns may help identify touch-points that could help minimize customer dissatisfaction with the claims process.

We used data from auto and property claims in our analysis. Upon completion of the claim process each customer filled out a satisfaction survey. In total, there are 166,504 claims in our data, of which 7,874 (4.73%) customers indicated that they were dissatisfied with the claims process. Of these, 109,031 claims belonged to the auto category (3.2% dissatisfaction), and 57,473 claims belonged to the property category (7.6% dissatisfaction).

There are a total of 236 event types in our data (232 for auto, 202 for property), 12,758,448 event instances (8,965,952 for auto, 3,792,496 for property) and, on average, 86 event instances/claim.

Both the auto and property datasets were split into 60/40 train/test sets. Then the algorithm was run on each training set separately. We used a p-value threshold of 0.01 and set the pruning cutoffs at 2.84% of the total event instances in each train set. For our experiments we allowed the cutoff for infrequent events and the cutoff for infrequent pairs be the same value. A smaller cut-off results in a substantially larger number of T-patterns, that increases the run-time for the algorithm exponentially. In addition, the KL divergence cutoff was varied between 0.01 and 0.000625. The primary benefit of the KL cutoff is that it helps identify the T-patterns that are more likely to discriminate between satisfied and dissatisfied customers.

Next, we used each discovered T-pattern as a binary feature to classify satisfied and dissatisfied customers. Using these features, we trained 4 different scikit-learn (Pedregosa et al. 2011) (version 0.18) models: Logistic Regression with L1 penalty (LogR L1), Logistic Regression with L2 penalty (LogR L2), Random Forest (RF; with 100 estimators, and out-of-bag sampling set to True), and Gradient Boosting Classifier (GB). Each algorithm used it's default parameter settings unless otherwise specified here. The best results were consistently obtained with GB so for the rest of the paper we will only report GB results. The training set is used to generate the T-Patterns. The binary features corresponding to these T-Patterns are used to build a model over the training set. The model is then used to predict satisfaction of claimants in the hold-out/test set.

Table 2: Auto & Property AUCs for Train/Test sets; supervised runs with varying KL cutoff parameter, best results shown at KL cutoff of 0.00125. Best performance in **bold**. *TP (= T-pattern). BOW (= Bag-Of-Words) features are basic counts of each event. We excluded adding any aggregate features (e.g. total number of events in claim, number of unique events in claim, duration of claim, etc.) for simplicity of comparison.*

Feature Set	**Algo**	**Auto Train**	**Auto Test**	**Prop Train**	**Prop Test**
TP	GB	0.744	0.714	0.720	0.678
BOW	GB	0.771	0.747	0.739	0.703
BOW+TP	GB	0.783	**0.753**	0.757	**0.721**
cSPADE	GB	0.757	0.523	-	-

We use a simple summary measure of binary precision/recall: Receiver Operating Characteristic (ROC) curves (Fawcett 2006), and report Area Under the ROC Curve (AUC) as our prediction measure. The prediction results for GB can be seen in Table 2. This shows that the presence/absence of these T-Patterns are a good indicator of satisfaction. Example T-Patterns can be seen in Table 3.

We compare our algorithm's result against a classic frequent sequence mining algorithm called cSPADE (Zaki 2000). We used the R package *aRulesSequences* to run the cSPADE code against our datasets (R Core Team 2018; Hahsler et al. 2018; Hahsler, Gruen, and Hornik 2005; Hahsler et al. 2011). This algorithm was chosen because it represents sequence mining that takes into account the order of events, but not the variable time between events - an aspect of our algorithm which we believe is distinguishing. This would allow for the closest possible match for the sake of comparison between algorithms and it would help illustrate why variable time between events is desirable to include. (Zaki 2000) states that the cSPADE algorithm can be run with a constraint to handle classes, and we chose to run the algorithm with this constraint so the results would be comparable to our supervised runs. The results can be seen in Table 2. Note that cSPADE cannot handle datasets with events that only occur in the testset but are never seen in the training set, hence we could not generate results for the property dataset.

The cSPADE algorithm was run using a support value of 0.6 which resulted in 31,642 sequences found after 4.3 hrs of run time on a single machine with 256 GB of RAM, and 20 physical CPU cores (note that this R session was one of multiple processes occupying the machine at runtime). Lower support values would error out for various reasons.

As was the case for T-pattern detection, we used the training set to generate the sequences. The binary features cor-

Table 3: Example T-patterns from Auto & Property

Data	T-patterns
Auto	(ClaimReported 1,7 (VehicleEdited 1,22 ClaimClosed))
Auto	(ClaimReported 1,28 ((Suspense 1,8 ClaimClosed) 1,56 grpPymtSubmit))
Prop	((ClaimSetup 1,14 ClaimClosed) 5,51 AssociatedToParty)
Prop	((ClaimSetup 1,57 (FYISent 1,7 Suspended Item)) 5,53 ClaimReopened)

1. The first example shows that changes to the vehicle description/condition were made 1 to 7 days after a claim is reported, and within 1 to 22 days after that the claim was closed. This is correlated with a bad assessment or a complicated claim.
2. The second example shows a claim that was reported and paid for after 1-2 months via a batch payment method.
3. The third example shows a claim setup and closure within 1 to 14 days, but associated to an internal handler 5 to 51 days after closure. This correlates with a complicated claim not going smoothly and/or swiftly.
4. The forth example shows a claim which was suspended for a long time during it's resolution process, and then reopened 5 to 53 days after it was closed. This is a pattern from a slow moving claim.

responding to the sequences was used to build a model over the training set, and the model was then used to predict satisfaction in the test set. As can be seen by the testset AUCs, cSPADE doesn't do as well with the classification task as our T-pattern algorithm. Clearly the order of events alone (in cSPADE) isn't as discriminative as when the time aspect is also taken into account (in T-patterns).

6.2 Fraud Detection

Insurance fraud is an act committed by a party, resulting in an unlawful gain from an insurance company. Parties who may commit fraud include insurance applicants, policyholders, third-party claimants, professionals rendering services during the claims process such as doctors, lawyers and chiropractors, or other parties associated with a claim such as a witness in an accident. The majority of insurance company respondents in a 2013 survey conducted by FICO estimate that fraud can cost between 5% and 20% of claim volume. Hence, insurance companies are investing in automated fraud detection solutions more so than ever before.

Automated insurance fraud detection solutions rely heavily on business rules derived by domain experts and features manually engineered by the data analyst. In some cases, arbitrary time cutoffs are selected as part of the feature engineering process. For instance, a claim on a recently bound policy is a red flag. However, "recent" is usually an arbitrary point selected by the analyst with input from the business expert. Our algorithm helps in the automation of hand engineered features, resulting in more robust and data-driven feature engineering.

In addition to automated feature engineering, there is desire by the Special Investigations Unit (SIU) for a white box solution where T-patterns provide information beyond simply flagging a claim. The sequences of events detected by the T-patterns provide the necessary information for SIU triage to make quicker hand-offs to the next line of defense, the SIU investigator. In addition, the sequences of events found can be used as information to ultimately support the SIU investigator in the mitigation of a claim. This stretches the value of the data beyond a simple model input or business rule.

The analysis in this paper was focused on private passenger auto (PPA) claims that were referred to, and reviewed by SIU, at the well-known insurance company mentioned previously. The primary motivation for this analysis was to help in reducing the false positives of claims already referred to SIU. SIU was receiving about 70% of their leads from a particular source. 89% of claims referred by this source were false positives. By ranking or prioritizing already referred claims, this can allow SIU triage to ignore claims of lower quality, while focusing their energy on those with higher quality. In addition, as discussed, T-patterns provide automation in feature engineering, and visibility into the types of sequences of events that are associated with fraudulent activity. This additional information can help SIU triage be more efficient in their review.

SIU referrals constitute roughly 1.5% of the total PPA claims population. In addition to claims information, the data includes policy and billing transactions, and claim history. Each claim had a binary label indicating fraudulent or no fraudulent activity associated with the claim.

There were a total of 35,230 claims in the dataset, with 317 event types represented, and an overall fraud signal of 12.46%. The data were split into 70/30 train/test sets. Similar to what we did for the last set of experiments with the claims data, we used a p-value threshold of 0.01. We kept the cutoff for infrequent events and the cutoff for infrequent pairs to be of the same value. In addition, the KL divergence cutoff was varied between 0.01 and 0.000625.

We then used each discovered T-pattern as a binary feature to classify fraud claims. We applied the same 4 models previously described. The training set was used to generate the T-Patterns, and build a model. The model is then used to predict if a claim was fraudulent or not on the hold-out/test set. The prediction results for the best algorithm (GB) can be seen in Table 4; we again used AUC as our prediction measure. Our experiments indicate that for a signal ratio like we have in this dataset, a KL cutoff of 0.00125 provided the best pattern set. Example T-Patterns can be seen in Table 5.

The results show that the human engineered features (H.E.F.) work best for this complex problem. But the T-patterns detected enhance the results when added to the mix of features used for modeling. They increased the testset AUC by 1.5%. Due to their interpretability, they can also be easily explained for business purposes. End-user evaluation is underway.

7 Summary and Conclusions

We have successfully applied a recently proposed scalable T-pattern algorithm (Arora, Fung, and Tunuguntla 2017) to

Table 4: Fraud AUCs for Test sets; supervised runs with varying KL cutoff parameter, best results shown at KL cutoffs mentioned. Best performance in **bold**. *H.E.F. = Human Engineered Features*

Feature Set	Algorithm	AUC
TP	GB	0.568
H.E.F.	GB	0.744
H.E.F.+TP	GB	**0.759**

Table 5: Example T-patterns from Fraud

T-patterns
(backdated policy change 1,86 (loss occurrence 1,3 loss setup))
(payment late 1,3 payment process)

1. The first example shows that the customer made a policy coverage change that they asked to be backdated (i.e. applicable from some day in the past vs. from the time of the request). Within 1 to 86 days later they experienced a loss, and 1 to 3 days after that they submitting a claim for that loss.
2. The second example shows a customer made a late payment which was processed within 1 to 3 days. It is known that customers that are late for their payments are more likely to submit fraudulent claims.

two relevant problems in the insurance industry. The patterns discovered not only provide a way to automatically extract features that can be used in a later stage for classification purposes, but also provide rules that can be easily interpreted by humans. We have also introduced the proposed T-Pattern algorithm to the A.I. community as a tool to consider when the task is to discover and understand patterns that occur in temporal data. Furthermore, as mentioned before, this algorithm is the only available choice (as far as we know) that considers both order and time between events, while being able to scale to address the demand of many present-day business problems.

In recent years, there have been significant advances related to the use of deep learning architectures for modeling temporal data. Specifically, recurrent neural networks (RNN) and long short term memory networks (LSTMs) are designed to learn temporal dependencies. Such methods work very well for a large variety of prediction problems, and are now used widely. However, there are two limitations of such approaches: (i) they need large amounts of labeled data to work and they can be difficult to train due to the enormous amount of parameters to be learned (this is typically the case with deep learning architectures in general), (ii) they use black-box models that produce excellent accuracy but the results from such models are difficult to understand and interpret by humans. Our proposed T-pattern approach provides interpretable results for temporal data.

References

Arora, N.; Fung, G.; and Tunuguntla, S. 2017. T-patterns in business. Unpublished Manuscript. Available at SSRN: https://ssrn.com/abstract=3066839 or http://dx.doi.org/10.2139/ssrn.3066839.

Batal, I.; Cooper, G. F.; Fradkin, D.; Harrison, J.; Moerchen, F.; and Hauskrecht, M. 2016. An efficient pattern mining approach for event detection in multivariate temporal data. *Knowledge and Information Systems* 46(1):115–150.

Borrie, A.; Jonsson, G. K.; and Magnusson, M. S. 2002. Temporal pattern analysis and its applicability in sport: an explanation and exemplar data. *Journal of sports sciences* 20(10):845–852.

Casarrubea, M.; Jonsson, G.; Faulisi, F.; Sorbera, F.; Di Giovanni, G.; Benigno, A.; Crescimanno, G.; and Magnusson, M. 2015. T-pattern analysis for the study of temporal structure of animal and human behavior: a comprehensive review. *Journal of Neuroscience Methods* 239:34–46.

Fawcett, T. 2006. An introduction to roc analysis. *Pattern Recogn. Lett.* 27(8):861–874.

Hahsler, M.; Chelluboina, S.; Hornik, K.; and Buchta, C. 2011. The arules r-package ecosystem: Analyzing interesting patterns from large transaction datasets. *Journal of Machine Learning Research* 12:1977–1981.

Hahsler, M.; Buchta, C.; Gruen, B.; and Hornik, K. 2018. *arules: Mining Association Rules and Frequent Itemsets*. R package version 1.6-1.

Hahsler, M.; Gruen, B.; and Hornik, K. 2005. arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software* 14(15):1–25.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics* 22(1):79–86.

Magnusson, M. S. 2000. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods* 32(1):93–110.

Mannila, H.; Toivonen, H.; and Inkeri Verkamo, A. 1997. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* 1(3):259–289.

Mooney, C. H., and Roddick, J. F. 2013. Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.* 45(2):19:1–19:39.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research* 12:2825–2830.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roddick, J. F., and Spiliopoulou, M. 2002. A survey of temporal knowledge discovery paradigms and methods. *IEEE Trans. on Knowl. and Data Eng.* 14(4):750–767.

Zaki, M. J. 2000. Sequences mining in categorical domains: Incorporating constraints. In *9th ACM International Conference on Information and Knowledge Management*.

Zaki, M. J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal* 42(1/2):31–60. special issue on Unsupervised Learning.