Temporally Coherent Role-Topic Models (TCRTM): Deinterlacing Overlapping Activity Patterns

Evgeniy Bart$^{(\boxtimes)}$, Bob Price, and John Hanley

Palo Alto Research Center, Palo Alto, USA
{bart,bprice,jhanley}@parc.com

Abstract. The Temporally Coherent Role-Topic Model (TCRTM) is a probabilistic graphical model for analyzing overlapping, loosely temporally structured activities in heterogeneous populations. Such structure appears in many domains where activities have temporal coherence, but no strong ordering. For instance, editing a PowerPoint presentation may involve opening files, typing text, and downloading images. These events occur together in time, but without fixed ordering or duration. Further, several different activities may overlap – the user might check email while editing the presentation. Finally, the user population has subgroups; for example, managers, salespeople and engineers have different activity distributions. TCRTM automatically infers an appropriate set of roles and activity types, and segments users' event streams into high-level activity instance descriptions. On two real-world datasets involving computer user monitoring and debit card transactions we show that TCRTM extracts semantically meaningful structure and improves hold-out perplexity score by a factor of five compared to standard models.

1 Introduction

Models of user activities can be used to improve productivity and enable new services across a wide variety of domains such as finance, personal assistants, health care, and many others. However, such modeling is very challenging due to the complexity and variations in activities. We present a new generative model whose structure uniquely exploits properties of user activity streams in order to build better models of behavior in realistic contexts. Because these behavior models are generative, they can be used for a variety of classification and prediction tasks ranging from predicting future user needs, to detecting organizational saboteurs, to connecting users with common interests.

Real-world event streams (such as financial transaction streams or computer event logs) exhibit several forms of complexity. First, the latent structure is non-obvious, because semantically meaningful activities often manifest via groups of observed events that may be large, heterogeneous, and include significant variations in composition, order, and size. For example, editing a PowerPoint presentation may involve opening files, typing text, downloading images, and saving

DOI: 10.1007/978-3-319-23525-7_24

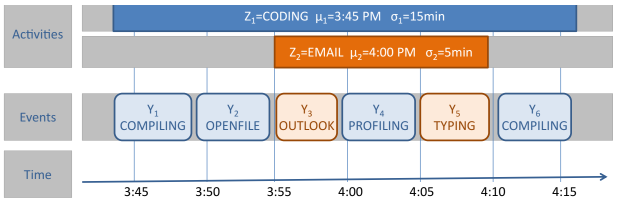


Fig. 1. A given user on a given day typically engages in multiple, possibly overlapping, activities. Each activity has a defined temporal extent. These activities in turn generate a sequence of events at specific times. Events from multiple activities may be interleaved.

files, but the exact order and frequency of these events varies significantly. Some users prefer saving their presentation more often than others; some presentations may involve a lot of text and not many images, while for others the opposite is true; some users download content using Firefox, while others prefer Chrome or Safari; and so on.

Second, the activities generating events can overlap in complex ways: a user email activity, a user powerpoint development activity and a background operating system update activity could all be active simultaneously. In real-world data streams, the events generated by these activities are intermixed in an extended history and are not segmented out or distinguished in any way.

Third, users typically comprise multiple distinct subgroups with very different behaviors. For instance, an office may include administrators, salespeople, and engineers; these will have very different computer activity distributions. Identifying these groups may be interesting in itself; in addition, attempting to create one model for all groups may result in poor performance.

As an example, consider the events associated with a workstation user who is coding and writing emails during the same interval (see Figure 1). These two activities result in interleaved operating system events such as compiling, opening of files and text entry. We note that while activities have temporal extents, the events generated by the activities do not have strong sequential orderings. Compiling, editing and file events are all part of coding, but the specific ordering of these is not highly determined.

A desirable goal is abstracting these complex raw event streams into a concise, high-level description of the user's typical activities. In this paper, we explain how our proposed model addresses the issues outlined above, and provide examples of its modeling capability on two challenging real-world datasets.

The remainder of this paper is organized as follows. In section 2, we summarize relevant prior work and highlight the differences from our proposed model. In section 3, we describe the proposed TCRTM model. The experiments in section 4 demonstrate that TCRTM successfully overcomes the challenges outlined above and significantly outperforms previously available methods. We conclude with some final remarks in section 5.

2 Related Work

Although some existing methods could be applied to analyze overlapping, loosely structured activities as described in this paper, none addresses all of the challenges outlined above. Below, we summarize the most relevant prior work, categorized by approach.

Handcrafted Models of Behavior. Behavior analysis has a long history in psychology and organizational theory [11], but these theories do not provide a formal computational model that can be used for prediction. The multi-agent systems community has applied computational agents to modeling of organizations [3]. These approaches are based on simulation, which allows us to see the implications of predefined behaviors. We find that our data changes too rapidly to admit handcrafted rules. Existing models from psychology and organizational behavior do not include an inferential component capable of directly extracting new behavioral insights from observational data.

Hidden Markov Models. Automatically learning human activity from observational data is a more recent endeavor but widely studied by many researchers. Many of these models employ variations on the Hidden Markov Model (HMM) [8]. Interestingly, in many transactional domains we find that the user activities have *temporal coherence* but do not have a strong sequential regularity. As illustrated in section 1, editing a PowerPoint presentation may involve events such as opening files, typing text, downloading images, and saving files, but the exact order and frequency of these events varies significantly from one case to another. Transitions between different activities are often not very structured as well; for example, some users may check email while working on a presentation, while others prefer to complete the current task before doing so. If such loosely structured activities were to be learned by an HMM, it would have to learn multiple different orderings of events and activities separately to account for all possible variations. This would combinatorially increase model complexity and training data requirements, and result in poor generalization.

Topic Models. Topic modeling is a key method for explaining discrete events in terms of a mixture of shared latent distributions. LDA [2] is perhaps the best-known example, although numerous generalizations exist (see below for some examples). Originally designed to simultaneously extract a set of topic vectors from a corpus of documents, and model the content of the documents as a mixture of extracted topics, LDA has been used in a wide variety of applications in which one wants to represent the content of the objects as a mixture of a small number of shared profiles.

A significant drawback of LDA is that it doesn't model time or temporal coherence. As a result, the topics it discovers may correspond not to a coherent user activity, but rather to a set of related events across multiple activities.

This is illustrated in section 4 and in Figures (3) and (4) giving examples of inferred activity types and LDA topics respectively. A detailed comparison of the proposed model to LDA is given in section 4.

Dynamic Topic Models. such as DTM [1] ignore local ordering of events (such as the order of words in a single document), and model how both the topics and mixture priors change over time (across multiple documents, typically spanning years). DTM could capture topic content changes such as the fact that articles about football have recently begun to include material on head injuries as well as capturing changes in preferences over topics such as a shift from football discussion topic to a cell-phone apps discussion topic over time. Variations of this model have been applied to to capture shopping behavior over time [5] and industrial chlorine sensor network streams [14]. The problems addressed by dynamic topic models are orthogonal to the problem discussed here; our focus is on exploiting local temporal coherence of events (within a single document) to improve the semantic meaning of extracted topics and achieve better fit to the data. The drift of these topics over time is not addressed in this paper, although that is certainly an interesting future direction. Returning to the example in Figure 1, we'd like our model to understand, for example, that the 'check email' activity started later than the 'coding' activity, rather than determine how the 'check email' activity evolved over 5 years.

Dynamic Processes Treated as Stationary Distribution. Topic models have been applied to activity recognition in video sequences [9]. Spatio-temporal interest points (small patches in time that capture visual and motion texture) are extracted from a short video segment. A topic model is used to find a small set of topics that explains the features extracted from a set of short video clips of various actions. The interpretability of the model can be increased by semi-supervised training in which the classes (identified with latent topics) are known and a subset of instances are labeled [15]. In cases where there are common behaviors and rare behaviors, the model can be augmented to share features between common and rare behaviors so that the model only has to model how rare behaviors are different from common behaviors [13] – reminiscent of hierarchical population model style inference when data is sparse. These models treat an entire clip as features drawn from a stationary distribution, so they will not work unless the data is already segmented into regions of coherent activity (which is the goal of this paper).

Encoding Time in Vocabulary Words. Topic models have been extended to include time by augmenting vocabulary words with time information [6]. In this model, the user's location behavior is categorized as being at home (H), work (W) or other (O). The sequence of locations sampled at 1/2 hours resolution is grouped into trigrams (e.g., HHH or WWO) which are then augmented with a "coarse day segment" number (early morning 0-7am=1, morning 7-11am =2, etc.)

to get segment-augmented trigrams such as HHH1 or HHH2. The augmented trigrams are fed into a generic author-topic model which learns that specific users have certain patterns such as being home early in the afternoon or going out in the evenings. This model does not have any notion of an activity independent of time – the observation distributions are directly coupled to coarse time segments. So it is not possible to learn about shopping behavior in the morning and use this to make inferences about a shopping trip planned in the afternoon.

Preprocessing via Topic Models for Dynamical Models. In some work, topic models are used as a preprocessing step for later stages of activity recognition [4]. Topic models can be used as part of a preprocessing step to compress or project high-dimensional signals down to a vector over a small set of topics. The topic indexes can then be fed to an HMM or classifier. These models combine the drawbacks of LDA and HMM: for the preprocessing step to succeed, LDA must extract relevant topics, which is difficult with loosely structured event streams (see section 4); while HMM at the subsequent stage will only succeed if the sequence of transitions between topics is highly structured.

Topic Models over Multiple Corpora. Practitioners have recognized that there may be distinct subpopulations which need to be modeled in different ways. Topic models have been extended to explicitly model the interests of authors [12]. Topic models have also be extended to cover multiple corpora over time [16] in order to expose commonalities and differences of different media over time. While these models capture subpopulations, they, like LDA, do not reflect temporal coherence.

Non-negative Matrix Factorization. Mixture models can be applied to data to pull out possibly overlapping subcomponents. Non-negative matrix factorization [7] has been a popular approach for factorizing data. The technique has been explicitly applied to signal separation [10]. While NNMF does utilize the idea of events being generated by mixtures, it doesn't exploit the temporal coherence of activities. As a result (just like with LDA, see above), the clusters extracted are not necessarily coherent activities, but just collections of events with similar properties. In addition, NNMF typically requires very structured input that can be organized in a matrix or a higher-dimensional tensor. This makes it difficult to apply to our data, where different users have a different number of events at different time points.

In summary, existing models do not handle the challenges of distinct subpopulations with loosely structured, temporally coherent activities found in many real-world datasets. In the next section, we develop a model that has elements of a mixture model but incorporates temporal coherence, subpopulations and a notion of activity instances to handle these challenges.

3 Model

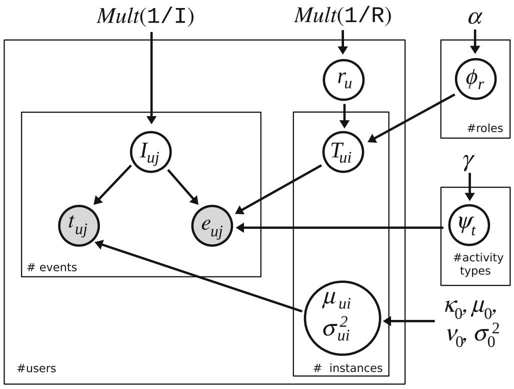
The observations that we are interested in modeling generally consist of streams of discrete events. Each event description includes the user who performed it, the timestamp, and an 'event type'. This event type is a discrete category label such as 'file open', 'image download', and so on.

Two key components of our model are *activity type* and *activity instance*. *Activity type* is a general category of activity performed by users, such as 'checking email' or 'road trip'. *Activity instance* is a specific exemplar of that activity being performed by a given user at a given time, such as 'user 124 checking email at 10 am' or 'customer 71 taking a road trip to Las Vegas on June 14th'. In our model, activity types are modeled as multinomial parameter vectors ψ, with ψ_t specifying the distribution over event types for the given activity type t. For instance, ψ_{email} might be a distribution favoring events such as 'selecting message', 'sending email', 'typing up a response', etc. (cf. Figure 3(a)). Activity instances are modeled by selecting an activity type, as well as the mean and variance of event timestamps in the activity instance. The mean and variance parameters define the temporal extent of the activity. So an 'email' activity instance might be for '10:00 am $\pm$ 10 minutes', and the email-related events around this time (as determined by a Gaussian density) will be preferentially associated with this activity instance. The smoothness of the Gaussian likelihood will facilitate the sampler's exploration of assignments of events to instances.

One additional component of the proposed model addresses the fact that users can often be grouped into distinct subgroups based on their observed activity. For example, company employees have different job roles. These roles dictate the types of activities users typically engage in. In our model, roles are modeled as multinomial parameter vectors ϕ. Each parameter ϕ_r specifies the distribution over activity types that users in role r are likely to engage in. A software engineer role might have a high probability for coding-related activities such as code compilation, whereas a marketing role might have a high probability for email and presentation activities.

The plate diagram for the TCRTM model incorporating user roles, activity types and activity instances is shown in Figure 2, where the corresponding generative process is also summarized.

The proposed TCRTM model could be compared to standard LDA [2] as follows. Consider treating users as documents, and individual events as words. Then both LDA and TCRTM explain the observed event types in a document via a set of topics ψ_t. The difference is that LDA has no concept of activity instance; all events assigned to a topic t are treated equally, and their timestamps are ignored. In contrast, in TCRTM an event is assigned to a 'topic' via an intermediate I_{uj} variable that corresponds to a specific activity instance; therefore, an activity type (such as 'checking email') may be repeated multiple times by each user, and there is an explicit separation of these multiple instances. The timestamps in TCRTM are not ignored, but rather are used to encourage temporal coherence of individual instances.



(a) Generative model

- For each activity type t, pick a multinomial distribution ψ_t over event types
- For each role r, pick a multinomial distribution ϕ_r over activity types
- For each user u:
  - Pick the role r_u for that user
  - For each of I activity instances:
    - Pick the activity type for the current activity instance from the role-specific distribution ϕ_{r_u}
    - Pick the mean and variance of timestamps for this activity instance
  - For each event j generated by the current user:
    - Pick an activity instance I_{uj}
    - Pick the event timestamp from instance-specific Gaussian distribution
    - Pick the event type from the activity for current activity instance

(b) Generative process

Fig. 2. (a): TCRTM plate diagram. Shaded nodes represent observable variables; variables not enclosed in circles represent hyperparameters. In the diagram, r_u is the role assigned to user u; ϕ_r is the distribution over activity types for role r; T_{ui} is the activity type for the i'th activity instance for user u; I_{uj} is the activity instance assigned to the j'th event of user u; e_{uj} is the event type and t_{uj} is the timestamp for the j'th event; ψ_t is the distribution of event types for activity type t; and μ_{ui} and σ^2_{ui} represent the time and duration of activity instance i. The conditional distributions are as follows: $\phi_r \sim \text{Dir}(\alpha)$; $r_u \sim \text{Mult}(1/R)$; $T_{ui} \sim \text{Mult}(\phi_{r_u})$; $I_{uj} \sim \text{Mult}(1/I)$; $e_{uj} \sim \text{Mult}(\psi_{T_{uI_{uj}}})$; $t_{uj} \sim N(\mu_{uI_{uj}}, \sigma^2_{uI_{uj}})$; $\mu_{ui}, \sigma^2_{ui} \sim NI\chi^2(\mu_0, \kappa_0, \nu_0, \sigma^2_0)$; $\psi_t \sim \text{Dir}(\gamma)$. R: number of roles; I: number of instances per user. (b): the corresponding generative process.

Compared to hidden Markov models (HMMs), the TCRTM allows multiple simultaneous activities to take place. Such multitasking is common in many datasets; for example, workstation data exhibits considerable overlapping activity due to both the user's attempts at multi-tasking, as well as due to the system executing background processes during the course of the user's normal work. These simultaneous activities are separated from each other in

TCRTM (a process called 'deinterlacing'), and are grouped into coherent activity instances. In contrast, in standard HMM implementations multiple simultaneous activities are usually modeled by augmenting the latent space to include a cross-product of multiple activities – a process that increases the modeling complexity significantly. The TCRTM also loosely models temporal coherence without imposing ordering. Unlike an HMM, the TCRTM's activity instances prefer representations in which the events of an activity occur close together in time without requiring any specific ordering of these events. To get the same generalization power as a TCRTM, an HMM must be trained on enough data to learn each possible ordering.

Finally, in addition to the differences discussed above, TCRTM also incorporates the concept of roles. These determine the activity types users can engage in, but are otherwise not constrained by official job titles. As a result, informal, but significant subgroups of people will be allocated distinct roles; these could correspond to different job types when modeling an organization's computer logs, or to customer groups when modeling debit card transaction data. The advantages of the TCRTM are summarized below:

- Compared to HMM, TCRTM can deinterlace overlapping activities. This is important for many practical datasets.
- Compared to LDA, TCRTM can deal with observable events that are temporally coherent (as opposed to activity that occurs throughout an interval of time)
- Compared to HMM, TCRTM can deal with observable events that are not strictly ordered.
- Compared to both LDA and HMM, TCRTM models user roles. This allows finding coherent groups of people with similar behavior.

3.1 Inference

The goal of inference is to estimate the parameters of the model given a collection of observed events and the hyperparameters (α, γ, κ_0, etc.). The parameters of interest describe the inferred domain structure (for example, ψ_t describes activity types in terms of event types that are likely under that activity), as well as specific assignments of objects to clusters (for example, I_{uj} represents the activity instance to which the j'th event of user u is assigned).

Our overall approach is to use Gibbs sampling, which allows drawing samples from the posterior distribution of the model's parameters given the data. The parameters of interest can then be estimated from these samples. For efficiency, we use a collapsed Gibbs sampler, where the variables ϕ, ψ, μ, and σ^2 are integrated out, and the remaining discrete variables r_u, T_{ui}, and I_{uj} are sampled until convergence. Estimates for the integrated-out variables can then be obtained in terms of the discrete variables.

The expressions below are derived using standard methodology for Gibbs sampling; therefore, the derivations are omitted. The resulting conditional distributions are shown for completeness, as well as for intuition and for comparison to standard models.

As usual, the sampling distributions are expressed using count data. In our notation, N represents a count variable. Its superscript indicates what entities are being counted (for example, N^I is a count of activity instances and N^e is a count of individual events). The subscripts are indices of the relevant entities. A dot in place of an index indicates summation over that index. The current entity being sampled is omitted from the counts.

In the conditional sampling distributions below, N^I_{rt} is the number of activity instances that belong to users with role r that have activity type t, excluding the current instance. Similarly, N^I_{ut} is the number of instances of user u that have activity type t, and T is the total number of activity types.

The conditional sampling distribution for the role of user u is:

$$p(r_u = r_0|\text{rest}) \propto \frac{\prod_t \Gamma(\alpha + N^I_{r_0 t} + N^I_{ut})}{\Gamma(\alpha T + N^I_{r_0\cdot} + N^I_{u\cdot})} \cdot \frac{\Gamma(\alpha T + N^I_{r_0\cdot})}{\prod_t \Gamma(\alpha + N^I_{r_0 t})}. \tag{1}$$

The conditional sampling distribution for the activity type of the i^{th} activity instance of user u is given next. Here, N^e_{te} is the number of events of type e assigned to activity type t, N^e_{uie} is the number of events of type e for user u assigned to activity instance i, and E is the total number of event types:

$$p(T_{ui} = t_0|\text{rest}) \propto \frac{\alpha + N^I_{r_u t_0}}{\alpha T + N^I_{r_u\cdot}} \cdot \frac{\prod_e \Gamma(\gamma + N^e_{t_0 e} + N^e_{uie})}{\Gamma(\gamma E + N^e_{t_0\cdot} + N^e_{ui\cdot})} \cdot \frac{\Gamma(\gamma E + N^e_{t_0\cdot})}{\prod_e \Gamma(\gamma + N^e_{t_0 e})} \tag{2}$$

The conditional sampler for the user u's j^{th} activity is given below. Here, t_ν is the Student's t distribution, and its parameters are $\nu_{ui_0} = \nu_0 + N^e_{ui_0\cdot}$, $\kappa_{ui_0} = \kappa_0 + N^e_{ui_0\cdot}$, $\mu_{ui_0} = \frac{\kappa_0}{\kappa_{ui_0}}\mu_0 + \frac{N^e_{ui_0\cdot}}{\kappa_{ui_0}}\bar{t}_{ui_0}$, and

$$\sigma^2_{ui_0} = \frac{1}{\nu_{ui_0}}\left[\nu_0\sigma_0^2 + SS_{ui_0} - N^e_{ui_0\cdot}\bar{t}^2_{ui_0} + \frac{\kappa_0 N^e_{ui_0\cdot}}{\kappa_0 + N^e_{ui_0\cdot}}(\bar{t}_{ui_0} - \mu_0)^2\right], \tag{3}$$

where $\bar{t}_{ui_0}$ is the empirical mean and SS_{ui_0} is the empirical sum of squares of timestamps for user u, activity instance i_0.

$$p(I_{uj} = i_0|\text{rest}) \propto \frac{\gamma + N^e_{T_{ui_0} e_{uj}}}{\gamma E + N^e_{T_{ui_0}\cdot}} \; t_{\nu_{ui_0}}\!\left(t_{uj} \mid \mu_{ui_0}, \frac{1 + \kappa_{ui_0}}{\kappa_{ui_0}}\sigma^2_{ui_0}\right). \tag{4}$$

It is interesting to compare these expressions with the corresponding sampling equation for regular LDA. In LDA, the probability of assigning an event e_{uj} to a topic z_0 is

$$p(z_{uj} = z_0|\text{rest}) \propto \frac{\gamma + N^e_{z_0 e_{uj}}}{\gamma E + N^e_{z_0\cdot}} \cdot \frac{\alpha + N^e_{uz_0}}{\alpha T + N^e_{u\cdot}}. \tag{5}$$

In TCRTM, the equivalent of topics is activity types. Events, however, are not assigned to activity types directly; rather, events are assigned to activity instances via eq. (4), and the activity instance i is associated with an activity type given by T_{ui}. Comparing eq. (5) to eq. (4), we note that the first term for LDA is similar to the first term for TCRTM, except z_0 is replaced with T_{ui_0} (since the activity instance i_0 has activity type T_{ui_0}, which is equivalent to the topic z_0 in LDA). The second term in LDA is absent from eq. (4), but appears instead as the first term in eq. (2), except that individual users u are replaced with user roles r that combine multiple users, and the fact that in TCRTM, activity instances are counted instead of individual events. Finally, the last term in eq. (4) is absent from the LDA sampling because LDA doesn't model event time stamps. This term simply encourages individual events from a particular activity instance to be clustered in time.

4 Experiments

We have experimented with two datasets. The first dataset includes debit card transactions from over 300,000 users over a period of approximately 7 month. The users are the beneficiaries of various state government programs; once a month each card is loaded with an allotment of money which the users can subsequently spend. The total number of transactions is about 50 million. Each transaction includes a timestamp and a merchant code. This merchant code is a description of the general type of products sold or services provided, such as “Veterinary services” or “Hardware stores”. This merchant code was used as the ‘event type’ in our model.

The second dataset includes data from monitoring user workstations at a large defense contractor. In this domain, the observables correspond to operating system primitives such as opening a file, executing a utility, or initiating a network connection. Each such primitive consists of two parts: the application that was used to perform the action (e. g., ‘firefox.exe’) and the action itself (e.g., ‘ImageDownloadEvent’). About 5000 employees were monitored over one month, resulting in over 100 million individual events.

TCRTM is not very sensitive to the choice of hyperparameters. For our experiments, the following settings were used: $\alpha = 1$, $\gamma = 1$. These were selected using simple logarithmic grid search. In addition, μ_0 was set to the empirical mean of all the timestamps in each dataset, and $\kappa_0 = 0.0001$ was used to reduce influence of the prior mean on timestamp variance (eq. (3)). Further, σ_0^2 was set to $(D/I)^2$, where D is the duration of the modeled time period and I is the number of activity instances within that time period. This prior simply splits the entire time period into I intervals of roughly equal length (note that the posterior distribution will adjust this prior based on the actual observed data). Finally, ν_0 was set to 5.0, again, chosen using simple logarithmic grid search.

TCRTM was initialized at random, and then Gibbs sampling was run for 200 iterations. Examining the marginal likelihood revealed that the sampler converged typically after about 30 iterations (not shown).

nlnotes.exe:EmailViewed	68%
notes2.exe:GenericTextEvent	8%
nlnotes.exe:EmailSent	5%

(a) ‘Checking email’

sshd.exe:FileReadEvent	39%
dropbox.exe:FileReadEvent	7%
httpd.exe:FileReadEvent	6%
searchprotocolhost.exe:FileReadEvent	3%

(b) ‘Using dropbox’

onenote.exe:FileReadEvent	26%
wmplayer.exe:FileReadEvent	26%
firefox.exe:OtherDownloadEvent	3%

(c) ‘Personal notes’

VRU BALANCE INQUIRY	46%
LOAD REGULAR DEPOSIT	10%
FINANCIAL INSTITUTIONS-AUTOMATED CASH DISBURSEMENT	9%
GROCERY STORES SUPERMARKETS	5%

(d) ‘Cash operations’

FAST FOOD RESTAURANTS (QUICK PAY SERVICE PILOT)	18%
GROCERY STORES SUPERMARKETS	15%
VRU BALANCE INQUIRY	10%
DISCOUNT STORES	6%
SERVICE STATIONS WITH OR WITHOUT ANCILLARY SERVICE	6%

(e) ‘Food’

RECORD SHOPS	9%
BOOK STORES	8%
DIRECT MARKETING-INBOUND TELEMARKETING MERCHANTS	8%
FAST FOOD RESTAURANTS (QUICK PAY SERVICE PILOT)	5%

(f) ‘Culture’

Fig. 3. Example activity types learned automatically by TCRTM. For each activity type, top event types and corresponding probabilities are shown; the numbers are rounded to nearest integer. The captions are not part of the model and were given by the authors for illustration. (a)-(c): workstation dataset. (d)-(f): debit card dataset. As can be seen, TCRTM successfully identifies semantically related groups of events.

The remaining parameters of interest in TCRTM are the number of roles R, the number of activity types T, and the number of activity instances per user I. For the debit card dataset, we’ve used $R = 25$, $T = 25$, and $I = 14$. R was chosen by trial and error, T was chosen by observing that for settings of $T > 25$ duplicate activity types started appearing, and I was chosen so that there would be roughly two activity instances per month (so that beginning-of-the-month and end-of-the-month spending patterns could be separated).

For the workstation dataset, we’ve used $R = 10$ and $T = 100$. Since activity types in this dataset (such as ‘checking email’ or ‘creating a presentation’) are

powerpnt.exe:ProcessStopped	29%
powerpnt.exe:ProcessStarted	29%
excel.exe:ProcessStarted	2%
excel.exe:ProcessStopped	2%
winword.exe:ProcessStarted	2%
winword.exe:ProcessStopped	2%

(a) ‘MS Office suite’

Fig. 4. Example topic learned by LDA on the workstation dataset. As can be seen, LDA grouped together a variety of events related to the MS Office suite. Although this grouping is understandable (as users who have Office installed typically use multiple applications within the suite), it is unlikely that they use all three applications simultaneously. Thus, the grouping reflects an artefact of software bundling rather than a semantically meaningful, coherent user activity.

of inherently much shorter average duration, we’ve used a setting of 10 activity instances per day, or 300 activity instances per month (which is our modeling period). While TCRTM could run with these settings as is, an additional observation is that activities are short and rarely span across day boundaries because of the way most people’s work days are scheduled. Therefore, we modified the model in Figure 2 to treat each day separately, with the corresponding obvious modifications to the sampling equations. The effect of this change is that instead of selecting one of 300 global activity instances for each event (most of the 300 with very low probability), the model only needs to select one of 10 activity instances for a particular day. This makes the sampler computationally more efficient.

TCRTM was compared to standard LDA [2]. LDA was chosen as a basis for comparison because it is naturally suited to modeling observations that are generated by several distinct latent processes, as the observations in our datasets are. Alternative methods, such as HMM, are unlikely to perform well on our datasets because there are no natural fixed transitions between events and between activities.

4.1 Activity Types

Several activity types discovered automatically by TCRTM are shown in Figure 3. As can be seen, the model organizes event types into semantically meaningful groups. Note that these groups encompass events that occur together when a user performs a natural task; they are not limited to grouping together all events from a single executable file or all events that co-occur in temporal proximity. For example, in Figure 3(a), all event types that occur when working with email are identified, even though they are performed by separate executable files. Note that the executable names themselves were not available to TCRTM; the events in question were encoded as ‘1733077456:98’, ‘2341822329:303’, and ‘1733077456:96’, providing no text similarity clues to the appropriate event

Table 1. Perplexity of LDA and TCRTM on the two datasets used. Lower values are better (note that perplexity measures the degree of surprise or confusion). Note that the values are on logarithmic scale. As can be seen, TCRTM significantly outperforms LDA on both datasets.

Method / Dataset	LDA	TCRTM
Debit cards	11.44	**2.56**
Workstation data	10.12	**5.91**

groupings. Similarly, multiple executable files pertaining to using the Dropbox service were grouped together (Figure 3(b)), and preference of several users to listen to music while editing notes with OneNote was identified (Figure 3(c)).

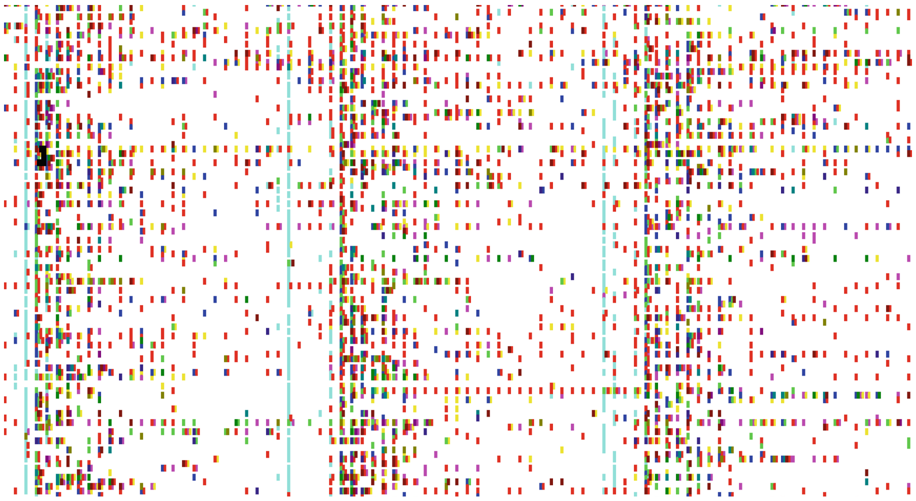
In contrast, topics learned by LDA often reflected not a coherent activity, but rather related events across multiple activities. This is illustrated in Figure 4. As can be seen, LDA grouped together a variety of events related to the MS Office suite. Although this grouping is understandable (as users who have Office installed typically use multiple applications within the suite), it is unlikely that they use all three applications simultaneously. Thus, the grouping reflects an artefact of software bundling rather than a semantically meaningful, coherent user activity.

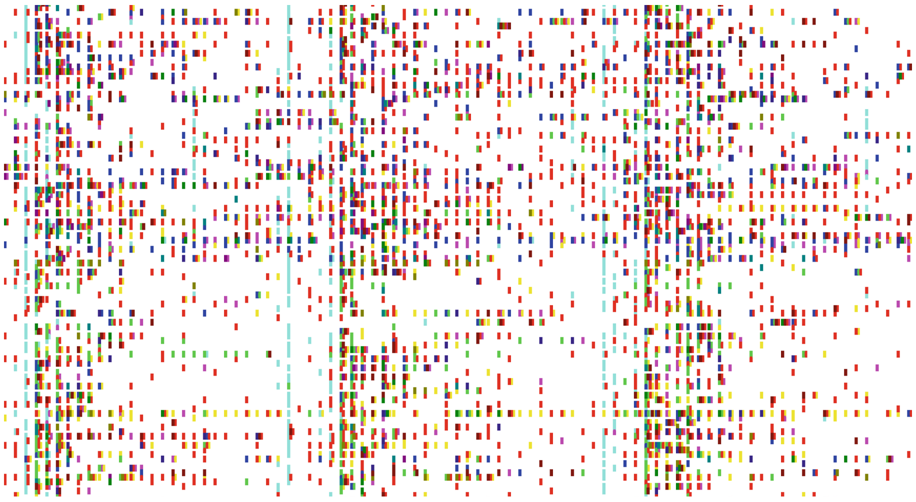
For debit card data, several interesting patterns were identified as well. For example, an activity type in Figure 3(d) groups together event types related to cash aspects of the card (checking the balance, receiving the monthly allotment, and withdrawing it as cash). Figure 3(e) shows an activity type related to buying food and other everyday items (such as gas).

The conclusion is that TCRTM can successfully identify semantically meaningful, coherent activity types.

4.2 Perplexity

Next, we compared TCRTM to LDA in terms of their ability to extract structure from data and anticipate future events. To perform the comparison, we split each dataset into a training and hold-out set (there was no overlap between the two sets). TCRTM and LDA were both fitted to the training set, and the perplexity of the corresponding models was then evaluated on the hold-out set. For this, the log-probability of each hold-out event was evaluated for each model under the parameters of an immediately preceding observed event. The results are reported in Table 1. As can be seen, TCRTM significantly outperforms LDA on both datasets.


(a) User histories in which each row of colored pixels denotes sequence of event types


(b) User histories ordered by inferred TCRTM roles

🟩 AUTOMATED CASH DISBURSEMENT	🟦 GROCERY STORES SUPERMARKETS
🟥 VRU BALANCE INQUIRY	🟪 FAST FOOD RESTAURANTS
🟦 LOAD REGULAR DEPOSIT	🟫 DISCOUNT STORES
🟩 EATING PLACES RESTAURANTS	🟪 DRUG STORES PHARMACIES
🟨 SERVICE STATIONS	

Fig. 5. Effect of TCRTM roles. (a): unordered histories; (b): histories ordered by TCRTM roles. In the top half of subfigure (b) we see *blue* online grocery purchases over the whole monthly cycle whereas in the bottom half of (b) we see a concentration of *bright green* ATM cash withdrawls early in the monthly cycle (cf. Figure 3(d)) and few online transactions of any type mid month. Thus TCRTM automatically infers cash-based vs. online client types. (Best viewed on-screen, enlarged and in color.)

4.3 Effect of Roles

To visualize the effect of modeling user roles, we performed an experiment on a small subset of the debit card dataset that contained a randomly selected set of 287 users and 50,000 transactions. The same settings as for the main dataset were used, except we reduced the number of roles to 2 (this was done for easier visualization, as well as due to the small size of the subset). The results are shown in Figure 5 (the figure is best viewed on-screen, enlarged and in color). In both images, each row corresponds to a different user. Time flows from left to right. Each transaction type is color-coded according to the legend at the bottom. Thus, each row shows a snapshot of user's behavior over 3 months (note that the full 7 months of data were used for modeling, but the display was truncated to 3 months due to space considerations). In Figure 5(a), the users are shown in the original, random order. In Figure 5(b), the same set of users was rearranged by their role, as inferred automatically by TCRTM. Thus, the only difference between sub-figures 5(a) and 5(b) is that in Figure 5(b), rows are shuffled such that users with the same role are clustered together. As can be seen, TCRTM groups users with similar behavior into the same role. There are noticeable similarities in behavior between users with the same role, and significant differences in behavior across roles. For example, the bottom part in Figure 5(b) contains more bright green transactions, corresponding to cash-based activities (cf. Figure 3(d)), and is overall brighter, while the top part contains more blue transactions and is overall darker. Examining the conditional distributions inferred for the roles indeed confirms that for the 'AUTOMATED CASH DISBURSEMENT' transactions (color-coded bright green), the probability under role 2 (corresponding to the bottom part in Figure 5(b)) is 0.19, while for role 1 (corresponding to the top part) it is only 0.08. For the 'GROCERY STORES SUPERMARKETS' transaction (color-coded dark blue), the probability under role 1 (top part) is 0.12, while under role 2 (bottom part) it is only 0.06. The conclusion is that the role modeling aspect of TCRTM identifies semantically meaningful groups of users.

5 Conclusions

The experiments comparing temporally coherent role-topic model (TCRTM) to conventional models such as LDA suggest that TCRTM can exploit the local temporal coherence of events and population subgroup modeling to increase the predictive power of the model. The significant increase in modeling power makes us optimistic about the potential for TCRTM to improve a broad range of applications related to activity analysis such as prediction, recommendation and classification. As such, we argue that the TCRTM is an important milestone that will stimulate more accurate algorithms for real-world transactional activity analysis applications.

Acknowledgments. The authors gratefully acknowledge support for this work from DARPA through the ADAMS (Anomaly Detection At Multiple Scales) program funded project GLAD-PC (Graph Learning for Anomaly Detection using Psychological Context). Any opinions, findings, and conclusions or recommendations in this material are those of the authors and do not necessarily reflect the views of the government funding agencies.

References

1. Blei, D.M., Lafferty, J.D.: Dynamic topic models. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 113–120. ACM (2006)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. In: NIPS, pp. 601–608 (2001)
3. Dignum, V. (ed.): Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models (2009)
4. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp 2008 (2008)
5. Iwata, T., Watanabe, S., Yamada, T., Ueda, N.: Topic tracking model for analyzing consumer purchase behavior. In: IJCAI (2009)
6. Gatica-Perez, D., Farrahi, K.: Discovering routines from large-scale human locations using probabilistic topic models. In: ACM Transactions on Intelligent Systems and Technology (2011)
7. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS, pp. 556–562. MIT Press (2000)
8. Natarajan, P., Nevatia, R.: Coupled hidden semi markov models for activity recognition. In: Proceedings of the IEEE Workshop on Motion and Video Computing, WMVC 2007, Washington, DC, USA, p. 10. IEEE Computer Society (2007)
9. Niebles, J.C., Wang, H., Li, F.-F.: Unsupervised learning of human action categories using spatial-temporal words. International Journal of Computer Vision **79**(3), 299–318 (2008)
10. Plumbley, M.: Conditions for non-negative independent component analysis. IEEE Signal Processing Letters **9**(6), 177–180 (2002)
11. Robbins, S.P.: Organizational Behavior: Concepts, Controversies and Applications, 5th edn. Prentice-Hall (1991)
12. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI 2004, Arlington, Virginia, USA, pp. 487–494. AUAI Press (2004)
13. Gong, S., Hospedales, T.M., Li, J.: Identifying rare and subtle behaviors: A weakly supervised joint topic model. In: Pattern Analysis and Machine Vision (2011)
14. Wei, X., Wang, X., Sun, J.: Dynamic mixture models for multiple time-series. In: IJCAI (2007)
15. Mori, G., Wang, Y.: Human action recognition by semilatent topic models. In: Pattern Analysis and Machine Intelligence (2009)
16. Zhang, J., Song, Y., Zhang, C., Liu, S.: Evolutionary hierarchical dirichlet processes for multiple correlated time-varying corpora. In: Proceedings of the 16th ACM (2010)