# Finding Progression Stages in Time-evolving Event Sequences

Jaewon Yang[†][*] Julian McAuley[†] Jure Leskovec[†] Paea LePendu[‡] Nigam Shah[‡]
[†] Computer Science, Stanford University, {jayang, jmcauley, jure}@cs.stanford.edu
[‡] Biomedical Informatics, Stanford University, {plependu, nigam}@stanford.edu

## ABSTRACT

Event sequences, such as patients' medical histories or users' sequences of product reviews, trace how individuals progress over time. Identifying common patterns, or progression stages, in such event sequences is a challenging task because not every individual follows the same evolutionary pattern, stages may have very different lengths, and individuals may progress at different rates.

In this paper, we develop a model-based method for discovering common progression stages in general event sequences. We develop a generative model in which each sequence belongs to a class, and sequences from a given class pass through a common set of stages, where each sequence evolves at its own rate. We then develop a scalable algorithm to infer classes of sequences, while also segmenting each sequence into a set of stages. We evaluate our method on event sequences, ranging from patients' medical histories to online news and navigational traces from the Web. The evaluation shows that our methodology can predict future events in a sequence, while also accurately inferring meaningful progression stages, and effectively grouping sequences based on common progression patterns. More generally, our methodology allows us to reason about how event sequences progress over time, by discovering patterns and categories of temporal evolution in large-scale datasets of events.

**Categories and Subject Descriptors:** H.2.8 [**Database Management**]: Database applications—*Data mining*

**Keywords:** User modeling, time series, event sequences

## 1. INTRODUCTION

A variety of natural processes generate sequences of data whose complex temporal dynamics need to be modeled. Such *event sequences*, in which individual entities generate a series of observations drawn from a finite categorical vocabulary, are ubiquitous in many applications. For example, an event sequence could represent a product purchasing history of an individual, a person's Internet browsing history, or a sequence of symptoms exhibited by a patient.

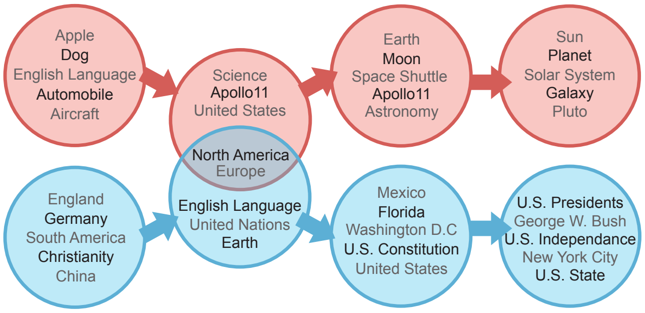


**Figure 1: Examples of the progression stages and the classes (Astronomy and U.S.A.) that we learn from Web navigation trajectories in the online game Wikispeedia. The top five most frequently visited pages are shown for each stage. Players start at some Wikipedia page and then move to the pages related to U.S. In the third stage, red players move towards astronomy-related pages, while blue players navigate towards U.S.-related topics. Both reach their corresponding goal pages in stage four.**

Event sequence data has two natural and interesting characteristics: The first is that sequences progress through distinct stages; the second is that there may be many different types or classes of progression.

*Progression stages:* Patterns of human behavior are generally not static as individuals evolve over time. Continuing the above examples, as users acquire more products, their tastes will change and thus, their preferences will go through a series of "expertise" levels [16]; or, as users search for information on the Web, their navigation strategies will progress through a series of phases [25]; finally, as a patient's disease progresses through various stages, they will exhibit different sets of symptoms. Understanding and modeling such sequences requires that we understand the mechanisms that cause them to change over time. In particular, sequences evolve through a series of progressing *stages* or *phases*. Other examples of progression sequences range from living cells that undergo various stages of mitosis to chess games that progress through several natural stages, like the opening, middle and the end of the game.

*Classes of progression:* In addition to understanding the various stages through which an individual sequence progresses, it is also necessary to *categorize* or *group* sequences according to how their temporal behavior evolves. For example, given a large set of product purchasing sequences of individuals, classes could represent groups of people that undergo similar evolution of their product purchasing patterns, *e.g.*, some users may gradually develop a taste for action movies, while others progress toward drama. Similarly, different patients may progress through stages of a disease differently depending on their age or gender [10]. Here classes could

[*] Current affiliation: LinkedIn

*WWW'14,* April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568044.

correspond to groups of people with a common disease or disease progression pattern.

Thus, in order to understand the temporal dynamics of event sequences it is crucial to solve two tasks. The first task requires us to identify different *stages* through which sequences progress and then *segment* individual sequences according to the discovered stages. The second tasks requires us to model different *categories* or *classes* of sequences. That is, to model classes of sequences that evolve according to different patterns of events. However, we have to consider both tasks simultaneously in order to better capture the diversity present in real sequence data.

Models for identifying progression stages are useful when solving a variety of tasks. Firstly, with richer models of temporal dynamics, we are better able to predict future events, such as the next product a person will consume, or the next symptom a patient will exhibit. Secondly, the stages and categories that we discover may themselves be meaningful. For instance, such models can help us to predict a patient's disease stage more accurately than is possible by examining their symptoms in isolation.

Modeling these types of temporal dynamics is a fundamentally difficult problem for a variety of reasons. Firstly, not every individual sequence evolves at the same rate. In addition, not every sequence will follow the same progression path or even progress through the same set of stages. Moreover, data may only be partially observed, *e.g.*, our first observation of a patient's symptoms may occur only after they already exhibit advanced symptoms. Finally, as there may be many different types of progression, sequences have to be both individually classified as well as segmented.

A variety of methods exist to model event sequences, though the above complications present issues for many existing models. Approaches based on mining frequently occurring subsequences [2, 18, 26] are not appropriate for this task, as the level of noise and large state-spaces mean that any specific pattern is extremely unlikely to appear repeatedly. Hidden Markov Models capture how latent states change [5, 7, 19, 22], though they typically assume that all sequences share the same set of latent states and thus progress in the same way. Other models of time-varying data, which are state-of-the-art for tasks such as movie recommendation on *Netflix* [9, 15], generally assume that users evolve according to a "global clock,", *i.e.*, their progression is tied to the calendar date. In contrast, modeling patient records (for example) requires that each individual progresses according to his or her own personal timescale. It is because of these above difficulties that a new model is required.

**Present work: Finding progression stages.** In this paper, we consider a broad definition of categorical *event sequences*: At a basic level, we model *ordered sets of events drawn from a finite vocabulary*. This level of generality allows us to model data from a variety of sources, including product reviews, browsing logs, media streams, and medical records.

We develop scalable methods to discover natural patterns of progression in time-evolving categorical sequence data. We achieve this by grouping sequences into different *classes*, based on common temporal patterns of events, while also individually *segmenting* sequences into automatically discovered progression stages. Both of these tasks are performed as part of a single optimization procedure, so that we simultaneously learn the categories and identify the progression stages of individual sequences. Our model is highly flexible in terms of how individual sequences progress—for instance, not every sequence needs to progress through every stage, and each individual sequence may progress through stages at a different rate; this flexibility is essential to capture the noise and variability present in real data.

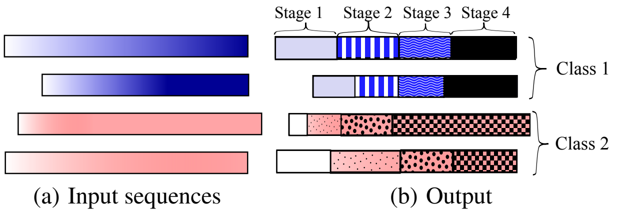


**Figure 2: Problem definition: Given input event sequences (a), we aim to categorize sequences into classes based on how they evolve, and we divide each sequence into progression stages (b).**

For example, Figure 1 illustrates the output of our algorithm when applied to human browsing traces on Wikipedia. We discover two sequence classes: people trying to reach pages about astronomy and people navigating towards U.S.-related pages. Simultaneously, we discover four stages of browsing behavior through which users progress when navigating towards a particular webpage.

More broadly, we apply our models to real-world event sequences from a variety of sources. We model people's consumption patterns on product review websites, such as *RateBeer.com*; we model people's browsing behavior using log data from *Wikispeedia* (a game that requires users to navigate Wikipedia pages [25]); and we apply our models to medical data of patients with chronic kidney disease.

In terms of experiments, we focus on three different aspects: predicting individual events, inferring progression stages, and grouping sequences into classes. For each aspect, we add qualitative analysis to show that our models help us to better understand and reason about the temporal dynamics of sequence data. First, we evaluate the ability of our method to predict future events in sequence data, *e.g.*, the next product a person will consume, the next page that she will navigate to, or the next symptom that she will exhibit. We observe that our method achieves a 30% gain in accuracy compared to existing methods for future-event prediction.

Second, we evaluate the accuracy and usefulness of the stages themselves, which we do by comparing them to known progression stages of patients with chronic kidney disease. The evaluation shows that our method can correctly estimate at what stage a symptom will appear, with a rank correlation higher than 0.8. We also analyze the stages that we infer in other datasets and observe that the speed at which a sequence progresses between stages signals the longevity of the sequence. For example, reviewers who advance too quickly or too slowly tend to produce fewer reviews in total compared to those who advance moderately.

Third, in our qualitative analysis, we find that the classes of sequences that we discover from navigation trajectories on Wikipedia correspond to different navigation strategies. We also discover that new users on product review websites initially consume similar products, before gradually "fanning out" and developing their own tastes, and then finally converging upon common subsets of products favored by "experts" in the community.

The remainder of this paper is organized as follows. In Section 2 we propose our model. Section 3 describes the data. Section 4 shows our experiments on event prediction, Section 5 discusses experiments on progression stages, and Section 6 presents experiments on classes of sequences. We discuss related work and conclude in Sections 7 and 8.

## 2. PROPOSED METHOD

Our goal in this paper is to discover the stages of progression that are common to a given set of event sequences. To achieve this goal, we develop a method based on a conceptual probabilistic model, which specifies how observed event sequences are generated from latent stages. We formulate the problem, develop the generative model, and then show how the latent stages of this model can be efficiently learned.

## 2.1 Problem Definition

We begin by defining the problem of finding progression stages. We assume that we are given a set of event sequences of different lengths, and we aim to infer their progression stages and classes. Our problem formulation is based on the following intuitions.

First, each event sequence progresses through a set of latent, discrete-valued "stages" over time, and observed events are generated depending on the sequence's current stage. Second, not only does each sequence have a different length, but the *duration of progression stages* for each sequence can be substantially different; some stages progress slowly, while others do so more quickly. Moreover, sequences may not progress through all stages, *i.e.*, they may start and finish at some intermediate stage.

The final intuition in our problem is that for any set of event sequences, there are multiple possible patterns of progression. To model different types of progression, we assume that there are latent classes or categories of event sequences, where sequences belonging to the same class progress through events in a similar way. We then aim to automatically "cluster" or "group" sequences to identify such common patterns of progression. In this way, we develop an unsupervised approach to clustering sequence progression data, by identifying sets of event sequences that follow common trajectories.

We formulate the problem of event sequence segmentation and classification as follows:

PROBLEM 1. *Given a set of event sequences, the problem of sequence segmentation and classification is to:*

- *find the class that each sequence belongs to; and*
- *assign each event to a stage, with stage assignments being non-decreasing over time.*

We illustrate the process in Fig. 2, and describe it in detail below.

## 2.2 Model Description

Here, we describe the generative process that we develop for modeling how observed event sequences are generated from a set of underlying latent progression stages.

We denote each event sequence by $x_i, i = 1, \dots, N$, and the $j$-th event of $x_i$ ($j$ ordered by time) by a categorical-valued $x_{ij} \in \{1, \dots, M\}$ where $N$ is the number of sequences and $M$ is the number of possible events. Each sequence may have a different length; we denote by $L$ the sum of the lengths of all sequences (*i.e.*, $L = \sum_i |x_i|$). We also assume that there are $C$ classes of sequences and that each class divided into $K$ stages. We assume for simplicity that all classes have $K$ stages, though our model can easily be adapted to accommodate a different number of stages per class.

Each sequence $x_i$ belongs to a single class $c_i \in \{1, \dots, C\}$. For each event $x_{ij} \in x_i$, we define $s_{ij} \in \{1, \dots, K\}$ to be the stage of the sequence $x_i$ at time $j$. Stages $s_{ij}$ are a non-decreasing function of time, *i.e.*, a sequence never progresses "backward".

$$\forall i, j, k \quad j \geq k \Rightarrow s_{ij} \geq s_{ik}. \tag{1}$$

From a modeling perspective, this constraint means that we capture patterns of temporal evolution that relate to the sequences of events, but are not tied to the exact time, or overall trends in the dataset. Also note that we do not require that any sequence $x_i$ should progress through all stages: some sequences may begin from intermediate stages, while some other sequences may end without reaching the last stage.

Given class $c_i$ and stages $s_{ij}$ for sequence $x_i$, we now specify how individual events $x_{ij}$ are generated. We employ a very simple generative process where $x_{ij}$ is independently drawn from a multinomial distribution with parameter $\Theta(c_i, s_{ij}) \in \mathbb{R}^M$:

$$x_{ij} \sim \text{Multinomial}(\Theta(c_i, s_{ij}))$$

Here, each $\Theta(c_i, s_{ij})$ represents separate distribution of events for a given stage $s_{ij}$ and class $c_i$. This way, we can ensure that sequences from the same class should have similar sets of events during the same stage.

Last, we also assume that $\Theta(c_i, s_{ij})$ is drawn from a uniform Dirichlet distribution with a hyperparameter $\lambda$:

$$\Theta(c_i, s_{ij}) \sim \text{Dirichlet}(\lambda).$$

Note that our approach can be generalized for generating $x_{ij}$ with more sophisticated models, *e.g.*, we could model $x_{ij}$ in each stage and each class using Latent Dirichlet Allocation (LDA) [4]. However, we found that our simple multinomial process works reliably in practice and allows us to fit the model very efficiently.

## 2.3 Inferring Progression Stages

We now explain how we can learn the stages of progression in event sequences based on our model. We are given a set of event sequences $\{x_i\}$. We also assume that we are given the number of classes $C$ and the number of stages $K$. (We will explain later how to estimate values for $C$ and $K$.) Our goal is to learn for each sequence $x_i$ the class membership $c_i$ of the sequence and the stage assignments $s_{ij}$ for each event $x_{ij}$ in the sequence. We achieve this by fitting the model, *i.e.*, we find classes $c_i$, stages $s_{ij}$, and multinomial distributions $\Theta = \{\Theta(p, q) | p = 1, ..., C, q = 1, ..., K\}$ by maximizing the log likelihood:

$$l(\Theta, \{c_i\}, \{s_{ij}\}; \{x_i\}) = \log P(\{x_i\} | \Theta, \{c_i\}, \{s_{ij}\})$$

Because variables $x_{ij}$ are conditionally independent of each other given $\{c_i\}, \{s_{ij}\}$, the log-likelihood becomes

$$\log P(\{x_i\} | \Theta, \{c_i\}, \{s_{ij}\}) = \sum_{i,j} \log P(x_{ij} | \Theta(c_i, s_{ij})).$$

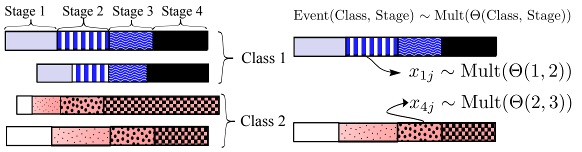
Thus, we aim to solve the following optimization problem:

$$\underset{\{c_i\}, \{s_{ij}\} \nearrow_j, \Theta}{\text{argmax}} \sum_{i,j} \log P(x_{ij} | \Theta(c_i, s_{ij})) \tag{2}$$

where $\{s_{ij}\} \nearrow_j$ is the monotonicity constraint in Eq. 1.

Optimizing Eq. 2 jointly for all sets of variables is highly challenging, since the problem is combinatorial and non-convex. We note that our formulation can be naturally cast in the framework of Expectation-Maximization (EM), where we compute soft assignments of the stages and the classes at one step, and then update $\Theta$ using these soft assignments. We note that we have experimented with the EM algorithm and found that EM converges prohibitively slowly. Thus we employ a coordinate ascent strategy, which is 1,000 times faster than EM in our experience, while yielding results of similar quality. Our coordinate ascent strategy is described below.

As illustrated in Figure 3, we iteratively update subsets of variables. First, we update $\Theta$ while keeping $\{c_i\}$ and $\{s_{ij}\}$ fixed (Fig. 3 (a)). Second, we update $\{c_i\}$ and $\{s_{ij}\}$ with $\Theta$ fixed (Fig. 3 (b)). We iterate these two steps until convergence, *i.e.*, until the classes and the stages that we learn do not change between successive iterations.


Stage 1
Stage 2
Stage 3
Stage 4
Class 1
Class 2
Event(Class, Stage) ~ Mult(Θ(Class, Stage))
$x_{1j} \sim \text{Mult}(\Theta(1,2))$
$x_{4j} \sim \text{Mult}(\Theta(2,3))$

(a) Updating stages (b) Updating $\Theta$

**Figure 3: Method description. (a) Update of stage and class assignments. (b) Update of model parameter $\Theta$ for each class and stage. We iterate the two steps until convergence.**

**Updating $\Theta$.** With stages $\{s_{ij}\}$ and classes $\{c_i\}$ fixed, we aim to find parameters $\Theta$ that maximize the log-likelihood $l(\Theta) = \log P(\{x_i\}|\Theta, \{c_i\}, \{s_{ij}\})$:

$$\underset{\Theta}{\text{argmax}}\, l(\Theta) = \log P(\{x_i\}|\Theta, \{c_i\}, \{s_{ij}\}).$$

Because variables $x_{ij}$ are conditionally independent of each other given $\{c_i\}, \{s_{ij}\}$, $l(\Theta)$ can be represented by summing log-probabilities for $x_{ij}$:

$$l(\Theta) = \sum_{i,j} \log P(x_{ij}|c_i, s_{ij}, \Theta) = \sum_{i,j} \log P(x_{ij}|\Theta(c_i, s_{ij})).$$

Note that $P(x_{ij}|\Theta(c_i, s_{ij}))$ does not depend on $\Theta(p,q)$ if $p \neq c_i$ or $q \neq s_{ij}$. Thus, $l(\Theta)$ is *separable* with respect to $\Theta(p,q)$:

$$l(\Theta) = \sum_{p=1}^{C} \sum_{q=1}^{K} l(\Theta(p,q)),$$

$$l(\Theta(p,q)) = \sum_{i,j} \mathbb{1}\{c_i = p \wedge s_{ij} = q\} \log P(x_{ij}|\Theta(p,q)),$$

where $\mathbb{1}$ denotes an indicator function. We find the optimal value of $\Theta(p,q)$ by maximizing $l(\Theta(p,q))$. Because $P(x_{ij}|\Theta(p,q))$ is a multinomial distribution, the optimal value of $\Theta(p,q)$ is the same as the empirical probability smoothed by the Dirichlet parameter $\lambda$:

$$\Theta(p,q)_r = \frac{\lambda + \sum_{i,j} \mathbb{1}\{c_i = p \wedge s_{ij} = q \wedge x_{ij} = r\}}{M\lambda + \sum_{i,j} \mathbb{1}\{c_i = p \wedge s_{ij} = q\}} \qquad r = 1, ..., M$$

**Updating stages.** Next, we describe how to update classes $\{c_i\}$ and stages $\{s_{ij}\}$ while keeping $\Theta$ fixed. This procedure means assigning each sequence $x_i$ to a class $c_i$ and assigning each event $x_{ij}$ to a stage $s_{ij}$, such that the log-likelihood for $x_i$ is maximized (subject to the monotonicity constraint of Eq. 1). We solve the following optimization problem for each sequence $x_i$:

$$\underset{c_i, \{s_{ij}\}\nearrow_j}{\text{argmax}} \sum_j \log P(x_{ij}|\Theta(c_i, s_{ij})). \tag{3}$$

To solve Eq. 3, we first compute the best assignment of stages (and the corresponding value of the maximized likelihood) for each value for class $c_i = 1, ..., C$. We then choose the class assignment that yields the highest likelihood. Thus, for each class $c_i$, we solve:

$$l(c_i) = \max_{\{s_{ij}\}\nearrow_j} \sum_j \log P(x_{ij}|\Theta(c_i, s_{ij})).$$

Optimizing $s_{ij}$ subject to a monotonicity constraint can be efficiently solved using dynamic programming via a transformation to the Longest Common Subsequence problem [3], whose complexity is bilinear in the number of stages and the number of events in the sequence. Then, we choose the optimal class $c_i$:

$$\underset{c_i}{\text{argmax}}\, l(c_i).$$

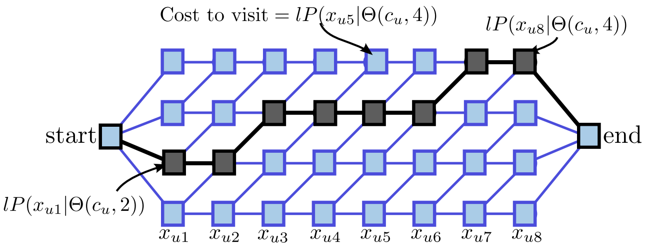

Cost to visit = $lP(x_{u5}|\Theta(c_u,4))$
$lP(x_{u8}|\Theta(c_u,4))$
start
end
$lP(x_{u1}|\Theta(c_u,2))$
$x_{u1}$
$x_{u2}$
$x_{u3}$
$x_{u4}$
$x_{u5}$
$x_{u6}$
$x_{u7}$
$x_{u8}$

**Figure 4: Our dynamic programming procedure for fitting stages. Each row represents a stage and columns represent events in a sequence. Given a particular sequence $u$ (columns; in this case, with eight events $x_{ui}$) we fit their optimal progression through four stages (rows) using dynamic programming; this is equivalent to finding the shortest path from the "start" to the "end" of the above graphs. This procedure is repeated for each class $c_u$ to choose the optimal class/path combination.**

Our dynamic programming procedure is depicted in Figure 4. Here we show the progression of a user with eight events (columns) through four stages (rows). The optimal path from the optimal class is used to choose the user's class label and progression sequence.

Next, we briefly describe the dynamic programming procedure for updating $s_{ij}$ for a given class $c_i$ (*i.e.*, finding the black path in Fig. 4). We compute the optimal cost $g(j,s)$ to reach $j$-th event at the $s$-th stage by forward recursion, from the fact that paths to $g(j,s)$ go through either $g(j-1, s-1)$ (*i.e.*, going up) or $g(j-1,s)$ (*i.e.*, staying at the same level):

$$g(j,s) = max(g(j-1,s) + \text{Cost}(j,s), g(j-1,s-1) + \text{Cost}(j,s)),$$

where $\text{Cost}(j,s) = \log P(x_{ij}|\Theta(c_i,s))$. When computing $g(j,s)$, we also record which action between "going up" or "staying level" is optimal. After computing $g(j,s)$ for all $j$ and $s$, we can find the optimal path by finding the optimal cost and action from the end of the sequence.

Last, we also note the complexity of our fitting algorithm. Updating stages for a given sequence $x_i$ and class $c_i$ requires computation linear in the number of stages and the number of events in the sequence ($O(K|x_i|)$). This means that updating stages and classes for all sequences requires $O(KLC)$ operations. Updating $\Theta$ has complexity of $O(L)$, which is negligible compared to $O(KLC)$. Thus, the complexity of one full iteration is $O(KLC)$, which is *linear* in the total number of events in the data. The code for our model is available at `http://snap.stanford.edu/snap`.

**Choosing the number of stages and classes.** When describing the model, we assumed that the number of classes $C$ and the number of stages $K$ are given a priori, which is true in some cases where domain knowledge may provide good estimates. In many cases, however, such domain knowledge may not be available. Thus, we provide a way to determine the number of classes and the number of stages automatically.

To automatically choose the number of stages and classes, we examine different values of $C$ and $K$, and choose values that maximize the goodness-of-fit via cross-validation likelihood. We split each sequence into 90% training set and 10% test sets $I_t$ uniformly at random, and then fit the classes and stages of the training part of the sequence. Then, for each event $x_{ij}$ in the test set $I_t$, we measure the probability of observing $x_{ij}$ assuming that it belongs to the same stage as its closest element from $x_i$ that appears in the training

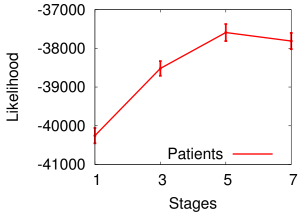


**Figure 5: Cross-validation likelihood (and standard deviation) versus the number of stages $K$ in the medical history of the patients with chronic kidney disease, when we fix the number of classes $C = 2$. The likelihood indicates that $K = 5$ is optimal.**

set, *i.e.*, we measure the following cross-validation likelihood:

$$\sum_{i,j \in I_t} \log P(x_{ij} | \Theta(c_i, \hat{s}_{ij}))$$

where $\hat{s}_{ij}$ is the stage of training event closest to $x_{ij}$.

**Algorithm initialization.** Before executing our algorithm, we must choose initial values for stage and class assignments. To initialize $\{c_i\}$, we divide sequences $x_i$ into $C$ different classes uniformly at random. To initialize $\{s_{ij}\}$, we split each sequence $x_i$ into $K$ stages at uniform intervals, *i.e.*, for each sequence $x_i$, we set $s_{ij} = 1$ for the first $|x_i|/K$ events, $s_{ij} = 2$ for the next $|x_i|/K$ events, and so on. Our method also includes a single hyper-parameter $\lambda$. We considered $\lambda \in \{1, 0.1, 0.01, 0.001\}$ and found that $\lambda = 1$ gives reliable performance across every dataset that we tried.

We note that our fitting procedure can be easily parallelized. Updating $\Theta$ can be done in parallel for each class and stage, and updating stages and classes can be parallelized for each sequence. Using parallelization with 20 threads, our model could be fit on our largest dataset (RateBeer) of 2 million total events within two minutes.

**EM algorithm.** Last, we briefly mention that we also experimented with an Expectation-Maximization (EM) procedure [20] to learn $\Theta$. Because $x_{ij}$ is generated from a multinomial distribution, the maximum likelihood estimate for $\Theta$ can be computed in closed form:

$$\Theta(p,q)_r = \frac{\lambda + \sum_{i,j} Q_{ij}(p,q) \mathbb{1}\{c_i = p \wedge s_{ij} = q \wedge x_{ij} = r\}}{M\lambda + \sum_{i,j} Q_{ij}(p,q) \mathbb{1}\{c_i = p \wedge s_{ij} = q\}}$$

where $Q_{ij}(p,q)$ is a posterior probability $P(c_i = p, s_{ij} = q | x_i)$ that $x_{ij}$ would belong to class $p$ and stage $q$. This posterior probability $P(c_i = p, s_{ij} = q | x_i)$ can be computed efficiently using the Forward-Backward algorithm [20].

We implemented the EM algorithm and compared it to our coordinate ascent method. The EM algorithm requires longer to converge, but it ultimately yields results similar to our coordinate ascent method. EM takes more than 1,000 times as long to execute. For example, it takes two days for EM to finish for the RateBeer dataset, whereas our method takes just two minutes. Thus, we focus on the coordinate ascent approach for the remainder of this paper.

## 3. DATASET DESCRIPTION

For our experiments, we consider five different time-evolving event sequences ranging from electronic medical records to online product reviews. We describe the datasets we consider and the definition of event sequences in each dataset. Table 1 provides the summary of our datasets.

**Product reviews.** First, we consider online product reviews from two large beer-reviewing communities (BeerAdvocate and RateBeer) [16]. These datasets contain all reviews from the inception of the sites (1998 and 2000, respectively) until 2011, containing 1,586,614 reviews from 33,387 users (BeerAdvocate), and 2,924,127 reviews from 29,265 users (RateBeer). We construct an event sequence for each user from the list of beers that they reviewed in chronological order. In this way, a sequence represents how users choose products (beers) as they develop their own taste and gain more experience. Since it is unlikely to be fruitful to model the progression of users who have rated only a few products, we discard users who have written fewer than 50 reviews. For a similar reason, we discard beers (which are individual events in our setting) that have been reviewed by fewer than 50 users. Overall, we consider 1,084,816 reviews from 4,432 users in BeerAdvocate, and 2,016,861 reviews from 4,584 users in RateBeer.

**Textual memes.** Our second dataset consists of quoted phrases in news articles and blog posts, provided using a system called NIFTY [23]. For each quoted phrase, NIFTY tracks which website posted an article quoting the phrase at what time. We take the quoted phrases from 2012, amounting to 2 million quoted phrases from 170,997 websites. The key idea in NIFTY is that a quoted phrase is a textual "meme", which represents the propagation of a very specific piece of information. We define a sequence to be a chronological list of the online media sources that mentioned a specific phrase, which represents how the meme spreads in online media space. In order to focus on memes that drew global attention and the role of important media sites, we only consider websites that have mentioned at least 0.5% of all phrases (10,000 phrases) and phrases that have been mentioned by at least 200 websites. This means that we consider 1,578,853 mentions for 4,866 phrases.

**Medical records.** Third, we consider electronic medical records of patients from Stanford Hospitals and Clinics, accessed via the Stanford Translational Research Integrated Database Environment repository [14]. The dataset spans 17 years with data on 1.8 million patients including 10.5 million clinical notes. We process the documents using methods described in [12] to create tuples of (medical term, patient, timeoffset). We consider patients who have been diagnosed with chronic kidney disease at least once. From medical terms corresponding to other disorders or symptoms mentioned in the records of these patients, we construct an event sequence of symptoms for each individual with a diagnosis of CKD. We focus on patients who have at least 50 medical terms in their history. Overall, we consider 393,334 terms from 1,835 patients.

**Web navigation traces.** Last, we consider Web navigation traces from the online game Wikispeedia [25], where players are given two random Wikipedia pages and must navigate from one to the other by clicking as few hyperlinks as possible. We regard each trace of a game (the Wikipedia pages that the player visited) as an individual sequence. In this way, sequences represent how a Web surfer navigates to reach a particular destination. We focus on game traces that have at least four pages and on pages that appear in at least 50 game traces, which results in a total of 164,308 page visits from 29,012 games.

Note that the progression stages in these datasets have different implications. In beer reviews, progression represents users gaining experience and developing their own taste [16]; in NIFTY, progression represents how information grows popular and then fades; and in patient data, it represents the development of diseases. Finally in Wikispeedia, progression represents how the players deploy different navigation strategies during different stages of browsing.

## 4. EXPERIMENTS ON EVENTS

Given sequences of events, our model can infer their underlying classes and the stages of progression. An example of our results is

| Dataset | Seq. | Event | $N$ | $L$ | $E(\lvert x_i \rvert)$ | $M$ |
|---|---|---|---|---|---|---|
| BeerAdvocate | User | Product | 4,432 | 1.1m | 244.8 | 5,161 |
| RateBeer | User | Product | 4,584 | 2.0m | 440.0 | 9,459 |
| NIFTY | Phrase | Media | 4,866 | 1.6m | 349.4 | 605 |
| Patients | Patient | Symptom | 1,835 | 0.4m | 214.3 | 124 |
| Wikispeedia | Player | Webpage | 29,012 | 0.2m | 5.7 | 1,048 |

**Table 1: The definition of sequences and events in the datasets and the data statistics. $N$: Number of sequences, $L$: Total number of events, $E(|x_i|)$: Average length (the number of events) in each sequences, $M$: Number of distinct events. $m$ denotes a million.**

shown in Fig. 8, where we show the most frequent events at each progression stage for two classes. Here, our model provides a summary of the progression of two classes of beer reviewers during three stages on BeerAdvocate.

In the next three sections, we perform experiments with our model. Each of the three sections focuses on three different aspects: individual events in the sequences, the progression stages that we learn, and the classes that we learn. In each of our experiments, we provide quantitative evaluation first and then analyze the results qualitatively. We will show that our model allows us to discover patterns and classes of temporal progression of online reviewers, information diffusion, Web navigation, and diseases.

The first experiment focuses on predicting missing events using our method. We formulate the task of predicting missing events in event sequences and evaluate the performance of our model quantitatively.

**Experimental setup.** To measure the accuracy of predicting missing events, we split each sequence into a training and a test set. We then fit the model using events from the training set and measure how accurately the method can predict the events that appear in the test set. Note that this can be seen as a multi-label prediction problem where $M$ distinct labels exist. We focus on the accuracy of predictions when we consider the $k$ most probable outcomes $\mathcal{O}_{ij}$, ($|\mathcal{O}_{ij}| = k$) for each missing event $x_{ij}$ in the test set $\mathcal{T}$

$$\frac{1}{|\mathcal{T}|} \sum_{i,j \in \mathcal{T}} \mathbb{1}(x_{ij} \in \mathcal{O}_{ij})$$

where $\mathbb{1}$ is an indicator function.

We employ two schemes to build our test sets. The first scheme is to consider the final (most recent) few events; this scheme evaluates the ability to predict future events in the sequences given events up to the present. The second scheme is to select a random sample of events from each sequence; this setting corresponds to the task of recovering missing events that may have happened in the past.

**Predicting events with our model.** We describe how we can predict the events in test set, *i.e.*, how to recommend the top $k$ items using our model. The idea is to infer the stage and class for each test event and then find the $k$ most likely items according to the corresponding multinomial distribution $\Theta$. Inferring the class is done using other training events in the same sequence. However, we cannot infer stages for events in the test set. Thus, for each test event, we assign it the stage of its chronologically nearest training event.

**Baselines.** We consider three baseline methods for multi-class prediction where we aim to predict the events in the test set given the training events.

First, we consider multi-class logistic regression, which aims to predict the label of missing events using the observed events as features. Whereas the training events in this problem contain just lists of events, logistic regression is a supervised method that requires training examples that have a "response variable" (label) and features. Thus, we divide the training events into "feature events" and "response events" so that logistic regression learns to predict the response events given the feature events. Among training events, we treat events adjacent to the test events as response events, and we treat the other training events as feature events. We then construct a feature vector $f_i \in \mathbb{R}^M$ using the feature events, and we learn $M$ logistic regression classifiers for each of $M$ distinct events.

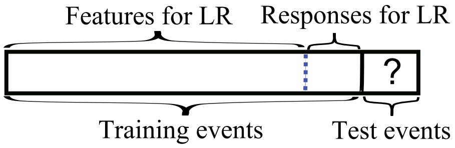


**Figure 6: As a baseline, we train a logistic classifier using training events. We split the training events into "feature events" and "response events" so that logistic regression learns to predict the response events given the feature events.**

$$f_{im} = |\{x_{ij} | x_{ij} = m, x_{ij} \in \text{feature events}\}|.$$

After learning the logistic regression classifiers, we aim to predict the test events. In this case, we treat all training events as feature events. We then pick the top $k$ labels based on the probability returned by logistic regression.

Our second baseline is a Hidden Markov Model (HMM), as an exemplar of models based on Dynamic Bayesian Networks (DBNs). After training the model, we estimate the latent state of the test event by choosing the state of the chronologically closest training event. Then, we generate $\mathcal{O}_{ij}$ using the $k$ most probable events from that estimated state. Comparisons against this method show how much benefit is obtained by modeling classes of sequences and by assuming that stages increase monotonically; without these additions, our model would be equivalent to an HMM.

Our third baseline method is a simpler version of our model where users progress at the same rate through the same set of stages. We call this baseline Model-U. We assume that all sequences belong to the same class, and for each sequence $x_i$, we set $s_{ij} = 1$ for the first $|x_i|/K$ events, $s_{ij} = 2$ for the next $|x_i|/K$ events, and so on. Using these stage assignments, we fit the parameter $\Theta$ for the multinomial distribution for each stage. Comparison against this model captures the effect of learning progression stages *individually* for each event sequence.

**Experimental results.** Table 2 shows the accuracy when predicting the final events in a sequence, where we output $k = 10$ most probable events for each test event (*i.e.*, $|\mathcal{O}_{ij}| = 10$). Among our three baselines, logistic regression consistently outperforms all the other baselines (Model-U and HMM) in all datasets. Thus, to conserve space, we show the performance of our method and logistic regression. The left two columns show absolute accuracies, while the third and fourth column show the relative improvement when we divide by the accuracy of randomly guessing one of $M$ values ($1/M$). The intuition behind relative improvement is that the overall difficulty of prediction depends on $M$, the number of possible event values. Even though the methods achieve low absolute accuracies in the beer data sets, our results here are significant as our method performs 100 times better than random guessing. Our method outperforms logistic regression on four datasets and achieves a relative gain of 130.7 on average, which is 32.4% higher than logistic regression whose average relative performance is 102.6. Note that unlike logistic regression, our method is not specifically designed for classification or prediction; nevertheless, the progression pattern learned by our model can provide a way to

| | Absolute Acc. | | Relative to random guessing | | Gain over baseline (%) |
|---|---|---|---|---|---|
| Method | Ours | Baseline | Ours | Baseline | |
| BeerAdvocate | 0.022 | 0.013 | 113.5 | 67.1 | 69.2 |
| RateBeer | 0.013 | 0.008 | 124.1 | 76.4 | 62.5 |
| Nifty | 0.338 | 0.297 | 204.5 | 179.7 | 13.8 |
| Patients | 0.563 | 0.608 | 69.8 | 75.4 | -7.4 |
| Wikispeedia | 0.135 | 0.109 | 141.5 | 114.2 | 23.9 |

**Table 2: Performance when predicting the most recent events of event sequences. Methods output the 10 most probable events. We compare to the performance of the best baseline (logistic regression).**

| | Absolute Acc. | | Relative to random guessing | | Gain over baseline (%) |
|---|---|---|---|---|---|
| Method | Ours | Baseline | Ours | Baseline | |
| BeerAdvocate | 0.030 | 0.014 | 154.8 | 72.3 | 114.3 |
| RateBeer | 0.022 | 0.009 | 210.1 | 85.9 | 144.4 |
| Nifty | 0.293 | 0.224 | 177.3 | 135.5 | 30.8 |
| Patients | 0.672 | 0.676 | 83.3 | 83.8 | -0.6 |
| Wikispeedia | 0.257 | 0.254 | 269.3 | 266.2 | 1.2 |

**Table 3: Performance of predicting a random set of missing events from event sequences. Methods output the 10 most probable events. We compare to the performance of the best baseline (logistic regression).**

predict the future events (or missing past events) of the sequences reliably. The only dataset where our model does not outperform all baselines is the Patients dataset. A possible explanation is that some common symptoms, such as "effusion", appear very frequently across all patients, and learning progression patterns would be less helpful for predicting such frequent symptoms.

Table 3 shows the performance when predicting a random sample of events. Again, our model outperforms the best baseline (logistic regression) in four datasets. For patient data, our method is on par with logistic regression. On average, our model yields a relative improvement of 179.0, which is 58% higher than what logistic regression achieves (128.7). Since our accuracy measure ignores how accurately we rank the top $k$ predictions, we tried other values of $k$ ($k \in \{1, 5, 20\}$) for evaluation, yet we found qualitatively similar results in comparing our method against baselines.

# 5. EXPERIMENTS ON STAGES

Our second set of experiments focuses on the stages of events that we infer from given event sequences.

## 5.1 Accuracy of Learning Stages

We begin by evaluating how accurately we can infer progression stages. For a set of events, we extract "ground-truth" labels for stages of particular events. We then measure how well the stages that we infer correspond to these ground-truth stages. Gathering information for such ground-truth stages is, in general, a challenging task; however, such information is available for medical events related to chronic kidney disease, which we study in this paper.

**Experimental setup.** Chronic kidney disease (CKD) has five stages, which are explicitly defined by the level of glomerular filtration. Our data contain explicit events about the CKD stage of patients, such as "chronic kidney disease stage $k$" ($k \in \{1, ..., 5\}$). Using such explicit events, we can estimate the ground-truth stage of other medical events (symptoms) by looking at the co-occurrence between the event and the "CKD stage $k$" events. For each symptom $e$ in our dataset, we measure the posterior probability $P_e(k)$ that the event "CKD stage $k$" happens with the event at the same visit. Then, we estimate the ground-truth stage $s_e^*$ of event $e$ by the posterior average value of $k$ (*i.e.*, $s_e^* = \sum_k kP_e(k)$). After estimating $s_e^*$, two researchers with a medical degree validated the values by manual inspection. The first two columns in Table 5 show a sample of four symptoms and their ground-truth stages.

| Score | Ours | Baseline |
|---|---|---|
| Kendall's $\tau$ | 0.810 | 0.659 |
| Pearson correlation | 0.447 | -0.007 |

**Table 4: Performance on learning the progression stages of chronic kidney disease.**

Given the training data, our model assigns each event to a specific stage. Thus, we compute the average value of stage assignments $\hat{s}_e$ for event $e$ (*i.e.*, $\hat{s}_e = E[s_{ij}|x_{ij} = e]$). We then compute the correspondence between ground-truth stage $s_e^*$ and the learned stage $\hat{s}_e$ using two standard metrics: Kendall's $\tau$ and the Pearson correlation coefficient.

**Baselines.** As a baseline, we consider Model-U that we considered in Section 4, where we segment each sequence into $K$ stages with the same duration.

To the best of our knowledge, we are not aware of existing methods that discover such *integer-valued* progression stages, which allow us to estimate at what point of progression a specific event would occur. Existing method for learning latent states [8, 19] estimate *categorical-valued* stages of events where no order between the stages exists.

**Experimental results.** Table 4 shows the performance of the methods. In both metrics, we show that our model outperforms the baseline Model-U, which shows that learning individual progression stages boosts the accuracy of inferring stages of events. Our method achieves a Kendall's $\tau$ (*i.e.*, rank correlation) of 0.8, which means that the stages learned by our model preserve the correct order for the more than 80% of the symptom pairs. Given that Model-U achieves $\tau = 0.659$, we achieve a relative improvement of 23%. In terms of Pearson correlation, the improvement over the baseline is even larger, as the stages learned by the baseline are negatively correlated with the true stages.

We further investigate the results of our model and Model-U. Table 5 gives a few examples of symptoms, their ground-truth stages, and the estimates by our model and the baseline. Note that our model's estimates match ground-truth stages much better than the baseline. For example, for secondary pulmonary hypertension, which, in practice, tends to happen at an early stage (stage 2), our model estimates a stage of 2.65 on average, whereas Model-U estimates a higher stage of 3.45. For hyperphosphatemia and acidosis, which happen at a later stage (stage 4), our model estimates the correct stage very closely (3.99 and 3.97 respectively), while the baseline estimates different stages, namely 3.71 and 3.21 (respectively).

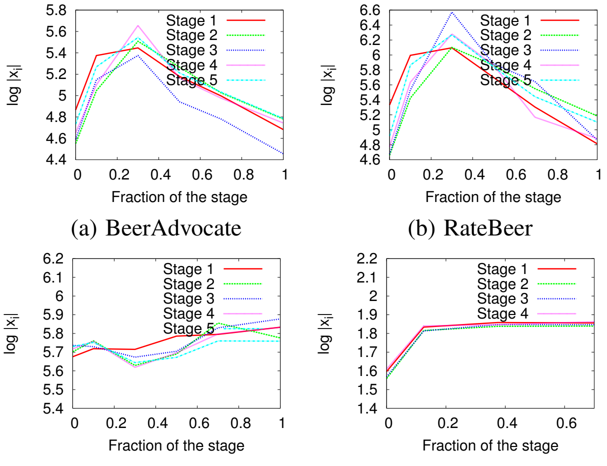
Given the poor performance of the baseline, we note that sequences can have a very different number of events at each stage, because diseases progress at different rates and within a disease individual patients progress at different rates. Our results show that our model can successfully learn the natural history of chronic kidney disease by correcting for such factors.

## 5.2 Relation between Stage and Sequence Length

We aim to gain some insight on how sequences evolve by analyzing the stages we learned qualitatively. In particular, we examine the relation between how quickly a sequence progresses and the length of the sequence. In other words, we ask the following ques-

| Symptom | Ground-truth | Ours | Baseline |
|---|---|---|---|
| secondary pulmonary hypertension | 2 | 2.65 | 3.45 |
| proteinuria | 3 | 3.19 | 2.94 |
| hyperphosphatemia | 4 | 3.99 | 3.71 |
| acidosis | 4 | 3.97 | 3.21 |

**Table 5: Examples of symptoms and their ground-truth stages. The stages predicted by our model and the baseline are also shown. Our method learns stages for symptoms more accurately than the baseline.**



(a) BeerAdvocate (b) RateBeer

(c) Nifty (d) Wikispeedia

**Figure 7: The average length (log-scale) of sequences as a function of the fraction of events elapsed at each stage.**

tion: Do sequences that get "stuck" at some stage tend to have more events? Or, do sequences that go through stages very quickly have more events? The length of sequence can be of great interest in many datasets; for example, it represents how actively a user enters reviews on BeerAdvocate and RateBeer, how popular a phrase is in NIFTY, or the skill of a player on Wikispeedia.

We examine the relation between the length of a sequence and the duration (measured by the number of events) that the sequence spends at each stage. For each sequence $x_i$, we measure the fraction of events $h_i(s)$ that the sequence has at stage $s$ ($h_i(s) = |\{x_{ij}|s_{ij} = s\}|/|x_i|$). In Fig. 7, we plot the average log-length of sequences $\log |x_i|$ as a function of $h_i(s)$ for each value of $s$.

In two product reviewing datasets (BeerAdvocate and RateBeer), the sequences have the maximum length at $h_i(s) = 0.25$. If users spend too long at a certain stage ($h_i(s) > 0.4$), or if they move to the next stage too quickly ($h_i(s) < 0.1$), they tend to produce fewer reviews. The users producing the most reviews are those who advance at a moderate rate. In Wikispeedia, we find an increasing relationship, meaning that players who advance more quickly will reach the target with fewer steps. In NIFTY, we see that the length of a sequence (the number of articles that mention a phrase) is not closely related to stage durations. We observed a similar pattern in the Patients dataset, yet we did not show the results because the length of a sequence (*i.e.*, the number of symptoms in the clinical notes) simply depends on how often the patient visited the hospital.

# 6. EXPERIMENTS ON CLASSES

Finally, we perform experiments in terms of the sequence categories that we learn. By inferring a class for each sequence, we essentially cluster sequences based on their common progression patterns. We quantitatively evaluate the quality of sequence clusters that we learn, and then we investigate the meaning of the classes in more detail.

|  | Absolute |  | Relative |  | Gain over baseline (%) |
|---|---|---|---|---|---|
| Score | Ours | K-Means | Ours | K-Means |  |
| Cluster quality | 0.161 | 0.103 | 9.5 | 6.1 | 56.3 |

**Table 6: Performance on clustering Wikispeedia game paths.**

## 6.1 Quality of Sequence Classes

We measure the quality of the sequence clusters by using a data-driven similarity metric between member sequences. For fair evaluation, we want to define such a similarity metric in terms of some external quantity, i.e., in terms of data with which our model is *not* provided. Among our datasets, only Wikispeedia provides such information, which allows us to measure sequence-sequence similarity.

**Experimental setup.** Given a class assignment $c_i$ for each sequence $x_i$, we measure the quality of each sequence class by computing the average pairwise similarity between its members [1]. In particular, we measure the quality $Q(p)$ of class $p$ as follows:

$$Q(p) = E_{c_i=c_j=p}[\text{Sim}(x_i, x_j)]$$

where $\text{Sim}(x_i, x_j)$ is a similarity function for a pair of sequences $x_i, x_j$. In Wikispeedia, we can measure taxonomical similarity between the pages using *Wikipedia categories* (*e.g.*, "Alligator" and "Crocodile" both belong to the category "Insects, Reptiles, and Fish"). It is natural to notice that games with destinations in the same category tend to be similar to one another, as players tend to navigate to similar intermediate pages. Therefore, we define $\text{Sim}(x_i, x_j)$ to be an indicator function that the last event (*i.e.*, the destination page) of $x_i$ and $x_j$ belong to the same category.

**Baseline.** We compare against the K-Means clustering algorithm using cosine distances. For each sequence $x_i$, we construct a feature vector $f_i \in \mathbb{R}^M$ by counting the occurrence of events:

$$f_{im} = |\{x_{ij}|x_{ij} = m\}|.$$

Then, we run K-Means clustering to cluster sequences where we use the cosine distance, $1 - \frac{f_i \cdot f_j}{||f_i|| ||f_j||}$, as a distance metric. K-Means will tend to group sequences with similar sets of events into the same cluster. The key difference between K-Means and our model is that our model considers the order of events, while K-Means ignores them.

We briefly note that we also considered bigram features (*i.e.*, features constructed by counting sequences of *two* events) for K-means. However, we found that K-Means with bigram features performed worse than K-Means with $f_i$ as defined above. We use the same number of clusters for K-Means as the number of classes used by our model.

**Experimental results.** Table 6 shows the average cluster quality of the K-Means clusters and our model's clusters (*i.e.*, classes). On average, our model's clusters achieve a cluster quality of 0.16, which is 56.3% higher than what K-Means clusters achieve (0.103). We also measure the relative gain over the average similarity between a random pair of sequences. Our model's clusters attain similarity values 9.5 times higher than what we would expect from random pairs of sequences.

## 6.2 Analysis on Classes

Our model allows us to learn "classes" of progression stages, each of which represents a specific pattern of how a particular group of sequences progress. We now investigate the classes of progression that we learn in more detail.

**Stage-wise similarity between classes.** We focus on the similarity between classes as stages progress. That is, we ask the following question: As sequences evolve, do the classes converge and have more homogeneous events? Or, do they diversify? To measure this, we conduct the following experiment. For each stage $s = 1, ..., K$, we measure the similarity between two classes $c_1, c_2$ by using the symmetrized cross entropy $H_s(c_1, c_2)$ [6] for events belonging to stage $s$:

$$H_s(c_1, c_2) = H'_s(c_1, c_2) + H'_s(c_2, c_1)$$

where $H'_s(c1, c2)$ is the asymmetric cross entropy:

$$H'_s(c_1, c_2) = E_{x_{ij}|c_i=c_2, s_{ij}=s}[-\log P(x_{ij}|\Theta(c_1, s))].$$

The cross entropy $H'_s(c_1, c_2)$ quantifies the uncertainty if we describe the events at stage $s$ in class $c_2$ using the multinomial distribution for class $c_1$ at the same stage $s$. The smaller it is, the more similar the two classes are to each other at stage $s$.

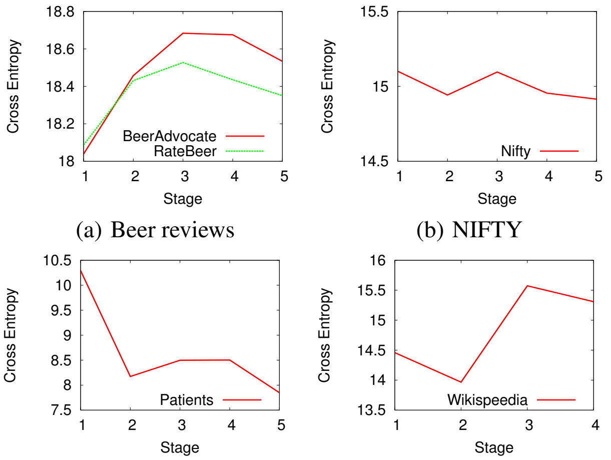
Fig. 9 shows the average cross entropy between classes at each stage. Fig. 9(a) shows that the entropy forms a bell-shaped curve whose maximum is at stage 3. Product reviewers begin with similar products, and then diverge from each other during stages 2, 3, and 4, where users develop their own taste. Finally, they arrive at similar sets of products that are favored by experts. Fig. 8 shows two classes that we learn in BeerAdvocate that follow this pattern. Fig. 8 shows the top seven most frequent products that we learn at stages 1, 3, and 5, where the classes have some overlap at stage 1, diverge at stage 3, and finally converge at stage 5.

On Wikispeedia, the cross entropy yields a minimum at stage 2 and then increases. High entropy at stage 1 is natural, as games begin from random starting points. The minimum cross entropy at stage 2 corresponds very well to the observation from existing literature [25] that players tend to navigate to a few "hubs" in their first move (*i.e.*, at their second page), before moving to more specific pages depending on their destination. Since players converge to hubs as their second page, stage 2 exhibits the minimum cross entropy. Then, game trajectories diversify depending on the topics of the destination pages.

The cross entropy pattern on Wikispeedia is clearly shown by the two classes in Fig. 1 (Sec. 1), which shows the five most frequent pages at each stage for two classes. Frequent pages at stage 1 are not similar to each other, as the games begin from a random page. At the second stage, players tend to move to "hubs", such as "North America" and "Europe." Then, red players move to "astronomy" pages, while blue players move toward "American" pages.

In the chronic kidney disease (CKD) patients data, the cross entropy tends to decrease as the stage increases. This suggests that patients show diverse symptoms other than CKD during its initial stages. However, patients tend to share similar CKD-specific symptoms as the disease develops. In NIFTY, the classes stay in parallel without converging or diverging.

**Classes in online media.** So far, we showed the classes that we learn on Wikispeedia (Fig. 1) and on product reviews (Fig. 8). We now investigate the classes of progression that we learn from the phrases quoted by online media in NIFTY. We examined the sequences (phrases) belonging to the same class and observed that the classes that we learn correspond to different topics, such as Politics, International, or Entertainment. Table 7 shows the top five most popular (frequently quoted) phrases in two of the classes that we learned in NIFTY. We can observe a clear distinction between phrases about entertainment (top) and political phrases (bottom).



(a) Beer reviews

(b) NIFTY

(c) Patients

(d) Wikispeedia

**Figure 9: Average cross entropy between the classes at each stage. The cross entropy shows stage-wise dissimilarity between the classes.**

| | |
|---|---|
| Class 1 | so devastating. we will always love you whitney, r.i.p<br>joker scene in dark knight rises<br>the daily show with jon stewart<br>the girl with the dragon tattoo<br>snow white and the huntsman |
| Class 2 | this is one small step for a man, one giant leap for mankind<br>they brought us whole binders full of women<br>the ecb is ready to do whatever it takes to preserve the euro<br>unchain wall street! they're gonna put y'all back in chains<br>evidence of calculation and deliberation |

**Table 7: Top five most popular phrases in two classes that we learn on the NIFTY dataset. The top class corresponds to phrases about Entertainment, and the bottom one contains political phrases.**

Our discovery in Table 7 suggests that political topics and cultural topics are mentioned by different media sites in a different order. We further examine this by finding the top five most frequent events (media sites) at each stage. Table 8 shows the results for stages 1, 3, and 5, and also shows that phrases about entertainment are first mentioned by independent media, then by broadcasting stations, and then finally by newspapers. On the other hand, political phrases are first quoted by newspapers, then by broadcasting stations, and lastly by forums.

**Classes of patients with chronic kidney disease.** We finally examine the classes that we learn from patients with chronic kidney disease (CKD). We identified two classes in this dataset, with the primary difference being the rate of occurance of albuminuria. In one class that we learn, albuminuria occurs extremely rarely, with probability $\Theta = 0.01\%$ in any stage, while in the other class albuminuria happens much more often ($\Theta = 0.05\%$, which is five times higher). Our findings correspond well with recent findings [10], which note that about 30% of CKD patients do not suffer from albuminuria contrary to the common belief that albuminuria is the hall-mark of screening for and early identification of CKD. In our analysis, the fraction of patients without albuminuria is 663 patients out of 1,835 (36%), which is similar to that reported in [10]. The natural history of CKD progression without albuminuria is relatively unknown, and is of active interest in nephrology because it comprises an injury pattern without classic glomerulosclerosis and

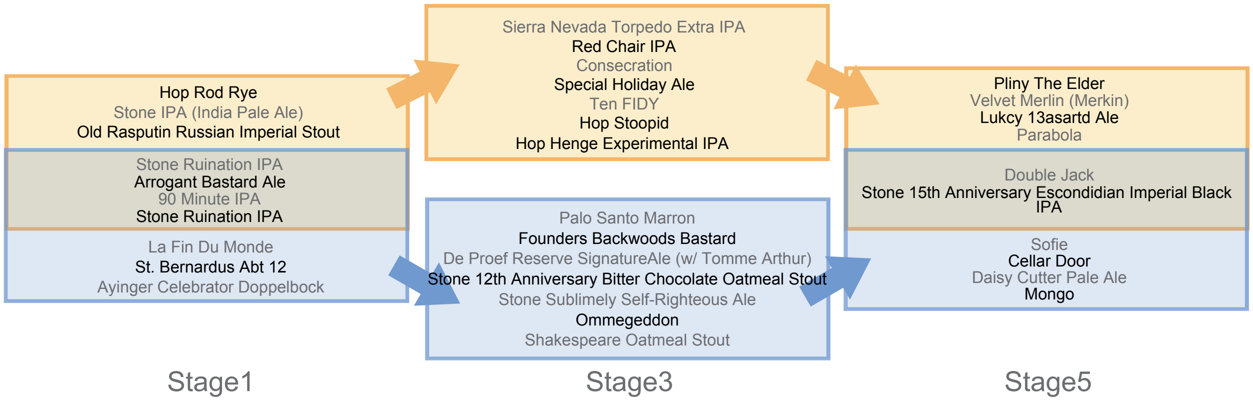

Hop Rod Rye
Stone IPA (India Pale Ale)
Old Rasputin Russian Imperial Stout
Stone Ruination IPA
Arrogant Bastard Ale
90 Minute IPA
Stone Ruination IPA
La Fin Du Monde
St. Bernardus Abt 12
Ayinger Celebrator Doppelbock
Sierra Nevada Torpedo Extra IPA
Red Chair IPA
Consecration
Special Holiday Ale
Ten FIDY
Hop Stoopid
Hop Henge Experimental IPA
Palo Santo Marron
Founders Backwoods Bastard
De Proef Reserve SignatureAle (w/ Tomme Arthur)
Stone 12th Anniversary Bitter Chocolate Oatmeal Stout
Stone Sublimely Self-Righteous Ale
Ommegeddon
Shakespeare Oatmeal Stout
Pliny The Elder
Velvet Merlin (Merkin)
Lukcy 13asartd Ale
Parabola
Double Jack
Stone 15th Anniversary Escondidian Imperial Black IPA
Sofie
Cellar Door
Daisy Cutter Pale Ale
Mongo
Stage1
Stage3
Stage5

**Figure 8: Top 7 most frequent products in two classes at progression stages 1, 3, and 5 from product reviews on BeerAdvocate.**

| | Initial stage | Middle (third) stage | Final stage |
|---|---|---|---|
| Class 1 | promiflash.de | news.yahoo.com | startribune.com |
| | thewrap.com | nbc.com | examiner.com |
| | huffingtonpost.com | abc.com | latimes.com |
| | examiner.com | entertainment.msn.com | news.yahoo.com |
| | wonderwall.msn.com | fox.com | entertainment.msn.com |
| Class 2 | news.yahoo.com | abc.com | townhall.com |
| | bing.com | nbc.com | freerepublic.com |
| | reuters.com | cbs.com | conservapedia.com |
| | guardian.co.uk | kwes.com | salon.com |
| | washingtonpost.com | fox.com | breitbart.com |

**Table 8: Top five most frequent sites at the initial, middle, and the final stages in the two classes of NIFTY quoted phrases in Table 7. Entertainment phrases (Class 1) tend to get mentioned by independent media sites during the initial stage, by TV stations during stage 3, before being mentioned by newspapers. On the other hand, political phrases are mentioned by newspapers during the first stage, then by TV during the middle stage, and finally by forums during the final stage.**

affects an estimated 0.3 million people [10]. We intend to analyze this patient group further in order to elucidate alternative makers that indicate progression via a non-albuminuria path.

## 7. RELATED WORK

Analyzing the progression of event sequences has been attempted in several different settings. One of the most notable approaches is that of episode mining [2, 11, 17, 18, 26], where one aims to find subsequences of events (episodes) that many sequences have in common. Because simply counting occurrences of subsequences may favor the most redundant ones, the task requires pruning techniques [2, 18], measures of subsequence importance [13, 26], or probabilistic modeling [11]. However, there are two drawbacks in these approaches [8, 24]. First, frequent subsequences focus on a very limited part of sequence data, as they do not tell us which events tend to happen *after* or *before* the chosen subsequences. Second, counting subsequences is susceptible to observation noise, which may result in partially observed or slightly permuted sequences. Rather than relying on counting, we apply a statistical approach to model whole event sequences, which allows us to summarize the global picture of sequences [8] while being robust to random permutations in the data.

The statistical model used in our approach is related to Hidden Markov Models (HMMs) [5, 7, 19, 22], which assume that observed sequential data arises due to a sequence of underlying latent states. HMMs have proven to be effective in a variety of applications, including time series clustering [22], event prediction [11], and speech recognition [19]. Whereas HMMs assume that any transition between latent states is possible, we enforce a specific structure on the transitions in which states are constrained to advance sequentially. We also introduce “classes” of stages so that sequences from different classes may evolve differently. We note that enforcing such a structure in state transitions is key to successfully capturing the progression of event sequences, and that HMMs without such structural constraints fail to model the types of data we consider in our evaluation (Section 4).

Further related work includes models of temporally varying matrices [9, 15]. For example, [9] considered time-varying bias terms to improve the accuracy of predicting movie ratings. In addition, [15] developed a multi-level tensor factorization approach to capture periodic trends in users’ Web-click behavior. However, these methods do not focus on the *individual development* of sequences (*i.e.*, users)—that is, how individuals evolve as they become more mature and gain more experience. In this work, we aim to consider such temporal aspects *individually* for each sequence.

## 8. CONCLUSION

In this paper, we developed a model to learn patterns of progression in time-evolving event sequences by grouping them based on how they evolve and by segmenting them into progression stages. Our method can process sequences with millions of events within a matter of minutes. Experiments show that our method can reliably predict the future events of sequences, accurately segment sequences into progression stages, and group sequences with similar properties into the same class. The progression stages and classes that we learn in real-world sequential data provide new insights on how product reviewers develop their own tastes when choosing products, how users navigate webpages, and how various topics are covered by different sources of online media.

There are also several avenues for future work. For example, it would be interesting to consider more sophisticated generative models of events [16]. On a similar note, allowing sequences to belong to multiple classes would be an interesting extension. Finally, our method discovers no structure among the stages of different classes, *i.e.*, each class evolves independently of the others; it would be interesting to explore whether the method can automatically find the overlaps between the stages of different classes. Adapting approaches recently proposed for extracting structure from online news might be particularly promising [21].

**Acknowledgements.** This research has been supported in part by NSF IIS-1016909, CNS-1010921, IIS-1149837, IIS-1159679, ARO MURI, ARL AHPCRC, Okawa Foundation, PayPal, Docomo, Boeing, Allyes, Volkswagen, and Alfred P. Sloan Foundation.

# 9. REFERENCES

[1] Y. Ahn, J. Bagrow, and S. Lehmann. Link communities reveal multi-scale complexity in networks. *Nature*, 2010.

[2] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. In *KDD*, 2012.

[3] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval*, 2000.

[4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Reserach*, 2003.

[5] E. Coviello, A. B. Chan, and G. R. G. Lanckriet. The variational hierarchical em algorithm for clustering hidden markov models. In *NIPS*, 2012.

[6] C. Danescu-Niculescu-Mizil, R. West, D. Jurafsky, J. Leskovec, and C. Potts. No country for old members: user lifecycle and linguistic change in online communities. In *WWW*, 2013.

[7] P. Felzenszwalb, D. Huttenlocher, and J. Kleinberg. Fast algorithms for large state space HMMs with applications to web usage analysis. In *NIPS*, 2003.

[8] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM Transaction on Knowledge Discovery from Data*, 2009.

[9] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 2010.

[10] H. Kramer, Q. Nguyen, G. Curhan, and C. Hsu. Renal insufficiency in the absence of albuminuria and retinopathy among adults with type 2 diabetes mellitus. *The Journal of the American Medical Association*, 2003.

[11] S. Laxman, V. Tankasali, and R. White. Stream prediction using a generative model based on frequent episodes in event sequences. In *KDD*, 2008.

[12] N. Leeper, A. Bauer-Mehren, S. Iyer, P. LePendu, C. Olson, and N. Shah. Practice-based evidence: Profiling the safety of cilostazol by text-mining of clinical notes. *PLoS ONE*, 2013.

[13] M. Liu and J. Qu. Mining high utility itemsets without candidate generation. In *CIKM*, 2012.

[14] H. P. Lowe H., Ferris T. and W. S. STRIDE–an integrated standards-based translational research informatics platform. *AMIA*, 2009.

[15] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa. Fast mining and forecasting of complex time-stamped events. In *KDD*, 2012.

[16] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*, 2013.

[17] D. Patnaik, S. Laxman, B. Chandramouli, and N. Ramakrishnan. Efficient episode mining of dynamic event streams. In *ICDM*, 2012.

[18] J. Pei, H. Wang, J. Liu, K. Wang, J. Wang, and P. Yu. Discovering frequent closed partial orders from strings. *IEEE Transactions on Knowledge and Data Engineering*, 2006.

[19] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989.

[20] S. Scott. Bayesian methods for hidden markov models. *Journal of the American Statistical Association*, 2002.

[21] D. Shahaf, J. Yang, C. Suen, J. Jacobs, H. Wang, and J. Leskovec. Information cartography: creating zoomable, large-scale maps of information. In *KDD*, 2013.

[22] P. Smyth. Clustering sequences with hidden markov models. In *NIPS*, 1997.

[23] C. Suen, S. Huang, C. Eksombatchai, R. Sosic, and J. Leskovec. Nifty: a system for large scale information flow tracking and clustering. In *WWW*, 2013.

[24] N. Tatti and J. Vreeken. The long and the short of it: summarising event sequences with serial episodes. In *KDD*, 2012.

[25] B. West and J. Leskovec. Human wayfinding in information networks. In *WWW*, 2012.

[26] C.-W. Wu, Y.-F. Lin, P. Yu, and V. Tseng. Mining high utility episodes in complex event sequences. In *KDD*, 2013.