Title
Modeling Events in Time using Cascades of Poisson Processes

Author
Simma, Aleksandr

Publication Date
2010

Peer reviewed|Thesis/dissertation

Modeling Events in Time using Cascades of Poisson Processes

by

Aleksandr Simma

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science
and the Designated Emphasis
in Communication, Computation, and Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael I. Jordan, Chair
Professor Dan Klein
Professor Cari Kaufman

Spring 2010

Modeling Events in Time using Cascades of Poisson Processes

Abstract

Modeling Events in Time using Cascades of Poisson Processes

by

Aleksandr Simma

Doctor of Philosophy in Computer Science

and the Designated Emphassis in Communication, Computation, and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Chair

For many applications, the data of interest can be best thought of as events – entities that occur at a particular moment in time, have features and may in turn trigger the occurrence of other events. This thesis presents techniques for modeling the temporal dynamics of events by making each event induce an inhomogeneous Poisson process of others following it. The collection of all events observed is taken to be a draw from the superposition of the induced Poisson processes, as well as a baseline process for some of the initial triggers. The magnitude and shape of the induced Poisson processes controls the number, timing and features of the triggered events. We provide techniques for parameterizing these processes and present efficient, scalable techniques for inference.

The framework is then applied to three different domains that demonstrate the power of the approach. First, we consider the problem of identifying dependencies in a computer network through passive observation and provide a technique based on hypothesis testing for accurately discovering interactions between machines. Then, we look at the relationships between Twitter messages about stocks, using the application as a test-bed to experiment with different parameterizations of induced processes. Finally, we apply these tools to build a model of the revision history of Wikipedia, identifying how the community propagates edits from a page to its neighbors and demonstrating the scalability of our approach to very large datasets.

Acknowledgments

During my time in Berkeley, I've had the pleasure and privilege of working in a wonderful research environment. I am very grateful to my advisor, Michael Jordan for allowing me to independently explore my interests, while providing useful advice when I needed it. His assistance and guidance has been a crucial part in enabling of my growth as a researcher. Additionally, Mike has assembled a wonderful group of students, who have shaped my grad school career for the better. I've learned much more through the random conversations that occurred in the lab than through most other channels. They, and my other student friends at Berkeley, taught me so much and kept me sane during times of adversity.

I would like to thank Moises Goldszmidt, whose guidance during my internship at Microsoft Research introduced me to many of the important questions in modeling events and inspired me on my choice of thesis topic. Discussions I had with him have been fundamental in shaping the initial ideas for this work and his personal enthusiasm for tackling important problems continues to inspire me.

Last but not least, I am immensely grateful to my parents, Fred and Galina, for raising me with a thirst for knowledge and supporting my path through life.

Contents

List of Figures

Chapter 1

Introduction

Time series data is extremely common, and the world is awash in data that are naturally represented as a marked point process. Any phenomenon that is composed of discrete, individual occurrences that happen at a particular instant can be thought of as a marked point process. However, the machine learning community has few tools for modeling this type of data without unnatural transformations such as discretization.

This work started with an applied question – given packet traces from a computer network, how could we build a statistical model of the network? The interesting, critical aspects of the data are the causal relationships and correlations between different packets that occur. While workload simulation has been considered in many contexts, for example by Bodnarchuk and Bunt [1991] and very recently by Ganapathi [2009], the task we are considering is the inverse problem of reconstructing the underlying structure and has not been sufficiently addressed.

While working to build a model of the network, some key assumptions for a reasonable model emerged:

1. Events are not independent. Furthermore, the relationship between events is the interesting aspect that must be captured.
2. An event may cause other events to occur after it, but not before. If A occurred before B, the only possible relationships are
   (a) A is unrelated to B
   (b) A caused/influenced B
   (c) Some other factor, occurring before both A and B influenced both of them.
3. The cause for an event is not necessarily the immediate event before it, but other things being equal, more recent causes are preferred.

The details of the model built for this task are described in Chapter 3. More generally however, a much broader category of interesting data sources produce events that follow these assumptions and can be modeled using ideas present in this thesis.

This body of work addresses the question of how these sorts of events can be modeled and presents a modular framework for building probability distributions over marked point processes. The resulting models can be used to compute the likelihood of a collection of events and make near-term predictions about expected future observations, but perhaps more importantly, the structure of the estimated model reveals facts about the observed process. These models enable us to determine what events can trigger a particular type of event and alternatively, they allow us to look back at observed data and infer what the possible triggers for an observed occurrence could be. They also allow a determination of the time scale on which dependencies occur and how the number of derived events depends on the attributes of the trigger.

Chapter 2 presents the building blocks for constructing models of events from several orthogonal components, each of which allows for design decisions and domain-specific engineering. The remainder of the work discusses applications to specific types of data that serve as case studies for further possible uses of these models. Chapter 3 applies a simple version to detecting dependencies in computer networks and uses hypothesis testing to identify interactions between machines. Chapter 4 constructs a model of Twitter messages about stocks and acts as a test-bed for studying the appropriate design choices in building these models. Chapter 5 applies the model to millions of Wikipedia revisions and demonstrates the scalability of this approach. Finally, Chapter 6 presents closing remarks and further research directions.

1.1 Related Work

The question of events and marked point processes has been previously considered by various communities and several known approaches exist. The field of queuing theory provides tools for building models of events arriving and being processed by one or more queues or a queuing network. In the computer networking and performance modeling community, as well as among queuing theorists, Markov-Modulated Poisson Processes [Rudemo, 1972, Fischer and Meier-Hellstern, 1993] are a way to model events. MMPPs are a Cox process, where the underlying random mean measure is a Markov chain. An alternative view of MMPPs is as a continuous-time hidden Markov model with binary emissions, as described in Wei et al. [2002]. More recently, Markov Poisson Cascades were proposed by Scott and Smyth [2003] as a model of web traffic.

Models similar to the ones proposed in Chapter 2 are well known in the statistics community. When an event causes a cascade of events of the same type, this is known as a self-exciting point process; when events can trigger events of different types, this is a mutually-exciting point process. Hawkes [1971] presents techniques for working with both of these processes

by characterizing their second moments and parameters are estimated by matching the covariance structure of the model to the observed data.

In the seismology community, trigger-based models, such as the one in Vere-Jones and Davies [1966] allow for randomly distributed "triggers" which then cause aftershock events as a Poisson process following the trigger. However, that class of models does not allow the aftershocks to then trigger successive aftershocks. On the other hand, the Epidemic Type Aftershock-Sequences (ETAS) model, introduced by Ogata [1988] uses a self-exciting point process and does allow aftershocks to cause successive aftershocks. The number of aftershocks expected after an earthquake event is modeled as a function of that earthquake's magnitude. In that work, the parameter estimation is handled by direct non-linear optimization of the log-likelihood.

More generally, point processes are a fundamental statistical concept and are covered in textbooks such as Cox and Isham [1980] and Daley and Vere-Jones [2003, 2008]. An excellent overview of the seismology work, comparing models fit by likelihood-based and second-moment based techniques, is Vere-Jones [2009].

On the machine learning side, the work on Continuous Time Bayesian Networks described in Nodelman et al. [2002, 2005] provides a version of Dynamic Bayes Nets that does not require time discretization. It is possible to build a model for events as transitions in the state of the CTBN: whenever a transition occurs, an event is taken to have happened. CTBNs are very computationally expensive due to the problem of entanglement – nodes that are conditionally independent for a particular time slice become entangled in a dynamic model and quickly become intractable. While Saria et al. [2005] enhances available inference algorithms for CTBN,s these models remain very computationally complex for the scale of problems we wish to approach and lack the capability to efficiently encode events with non-trivial feature spaces.

A graphical model formalism for Poisson Networks is presented by Rajaram et al. [2005] and is perhaps the closest in spirit to what is described in this thesis. Those authors describe a formalism for Poisson networks, in which there exist different nodes, representing different event types connected by directed edges. The rate of events at each node is a Poisson process, with the intensity function parameterized as a GLM of the empirical intensity of its parents within a fixed-lag window. However, such models only work for small, discrete types of events.

Finally, Wingate et al. [2009] present a recent nonparametric Bayes approach for learning timeseries structure called the Infinite Latent Events Model. The ILEM is a technique for building distributions over state sequences in discrete time, where on each time step, the model samples a collection of events (features) that were active in that time step from an from an infinite set of possible events so the latent state can be represented as an infinite 0/1 matrix. Each event that is active at time step t may cause zero or more events to be active at time $t+1$ through a Dirichlet process, leading to the effect that an event is more likely to cause events of the type it has caused in the past but has a non-zero probability of causing a new type of event. Each feature active at time t generates some collection of events for time

$t+1$, the union of which forms the events active at $t+1$. All this occurs in the latent event space, with the observed data at a timestep generated through an observation model from the binary latent state at that time. The Infinite Latent Events Models and the techniques described in this work are similar in the idea that each event causes more events later in time, with each event type having a tendency to generate a certain type of descendants. In our work, however, events take place in continuous time and cause successor events not just in the next timestep but at a later point in the future.

Chapter 2

Building the Model

In this section, we describe the models used for the remainder of this work. Since the models are composed of superpositions of Poisson processes, the first section provides an overview of well-known classical facts about Poisson processes and establish notation. Sections 2 and 3 describe how the models can be built from the building blocks of a fertility model, delay distribution and a transition distribution. Section 4 discusses options for constructing the fertility model and provides approaches for combining multiple fertility models. Finally, Section 5 discusses the computational aspects and provides accelerated algorithms for special cases of the model.

2.1 Poisson Processes

As background, this section will discuss Poisson processes, as they will appear repeatedly throughout this work. The Poisson process is an elementary point process that will serve as a building block for the more complex models described later in this work. This section uses the notation of Kingman [1993], which also contains further details and proofs of the results.

Definition 1. Let $(S, \mathcal{A}, \mu)$ be a measure space where μ is σ-finite and does not contain atoms[1]. Let Π be a random countable subset of S. For convenience of notation, we sometimes treat Π as the corresponding counting measure where $\Pi(A)$ is taken to mean $|\Pi \cap A|$. We say that Π is a draw from the Poisson process with intensity μ and write $\Pi \sim PP(\mu)$ when

1. $\Pi(A) \sim \text{Poisson}(\mu(A))$ for all $A \in \mathcal{A}$
2. For any non-intersecting $A_1, A_2, \ldots, A_n \in \mathcal{A}$, the values $\Pi(A_1), \Pi(A_2), \ldots, \Pi(A_n)$ are independent.

[1] A more general definition of a Poisson process allows for atoms but requires Π to be a multiset instead. As this will not be required for the remainder of this work, we will use the simpler definition.

There are many other ways of characterizing the Poisson process; for example, through Rényi's Theorem[Rényi, 1967], the assumption 1 above can be replaced by the assumption that $P(\Pi(A) = 0) = \exp(-\mu(A))$. The Poisson process emerges naturally from such definitions because it is a very fundamental mathematical object and a natural form for any counting process with independent increments.

Note that the Poisson process can be thought of either in terms of the locations of its specific points or as a counting measure[2]. The two are related by

$$\Pi_{\text{measure}}(A) \;=\; \sum_{p \in \Pi_{\text{points}}} \mathcal{I}(p \in A).$$

In many cases, it is useful to associate each point with an additional label called a mark.

Definition 2. Consider a measurable product space $(S \times T,\ \mathcal{A} \times \mathcal{B},\ \mu)$ where $\forall A \in \mathcal{A}$, $\mu(A \times T) < \infty$ and μ has no atoms. We will call a Poisson process on this spaces a Marked Poisson process where for each point (s, t), s is taken to represent the location of the event and t the associated features.

2.1.1 Useful Properties of Poisson Processes

These well-known properties of the Poisson process are proved in Kingman [1993].

Superposition: The sum of independent Poisson random variables is Poisson, that is for $x_i \overset{\text{ind}}{\sim} \text{Poisson}(\lambda_i)$, $\sum_i x_i \sim \text{Poisson}\left(\sum_i \lambda_i\right)$. Given a sequence of independent Poisson processes defined on the same space, $\Pi_i \overset{\text{ind}}{\sim} \text{PP}(\mu_i)$, it follows that for any set $A \in \mathcal{A}$, $\sum \Pi_i(A) \sim \text{Poisson}\left(\sum \mu_i(A)\right)$. More generally, the superposition of the processes $\cup_i \Pi_i$ is distributed as a Poisson process with the mean measure $\sum_i \mu_i$.

Mapping. Poisson processes can be transformed by mapping from one space to another. Since the mean measure for a Poisson process must not contain any atoms, it's important that the transformation not create any atoms in the induced mean measure; however, as long as this is the case, the mapped process remains a Poisson process.

Theorem 3. *Let Π be a Poisson process with σ-finite mean measure μ on the state space S, and let $f : S \to T$ be a measurable function such that the induced measure has no atoms. Then $f(\Pi)$ is a Poisson process on T having the $\mu^\star$, the measure induced by f, as its mean measure [Kingman, 1993].*

[2]Converting from the counting measure to the individual point locations requires that the σ-algebra be rich enough for all individual points in S be measurable.

A special case of mapping is projection. Let Π be a realization of a Poisson process defined on a product space $S = T \times U$ with σ-algebra $\mathcal{T} \times \mathcal{U}$ and mean measure μ. As long as for every element $t \in T$, the induced measure has no atoms, that is $\mu(\{t\} \times U) = 0$, the projection function $f = (t \times u) \to t$ defines a $f(\Pi) \sim PP(\mu^\star)$ where $\mu^\star(A) = \mu(A \times U)$ $\forall A \in \mathcal{T}$.

Extension (Marking, Coloring): Just as it's possible to project away some dimensions of a Poisson process, it's also possible to extend a Poisson process. Suppose we have a random Poisson process Π on S with mean measure μ_S, but we want to sample a process on $S \times T$ with the product measure $\mu_S \times \mu_T$. This is referred to as a marked Poisson process with marking space T. We can extend Π by, for every point $s \in \Pi$ sampling a $t_s \sim \mu_T$ and the resulting $\{s \times t_s\}$ will be a PP on $\mu_S \times \mu_T$.

If the measure on $S \times T$ we wish to sample from is not a simple product measure, s_t must be sampled from the conditional distribution $P_\mu(t|s)$.

Restriction: If Π is a Poisson process on $(S, \mathcal{A}, \mu)$ and $S' \in \mathcal{A}$, the restriction of Π to S' is a Poisson process on $\mu \cap S'$; this can be easily seen by considering the number of points falling into any set $A \in \mathcal{A}$ and noting it has the right distribution.

Thinning: Consider a $\Pi \sim PP(\mu)$. For each point in Π, discard it with probability $1 - p$ and keep it with probability p. So, with $c_x \overset{\text{iid}}{\sim} \text{Bernoulli}(p)$ for some $0 \leq p \leq 1$, define $\Pi^\star = \{x : x \in \Pi, c_x = 1\}$ which is a Poisson process with mean $p\mu$. This can be easily seen by letting Π' be an extension of Π from S to $S \times (0, 1)$ with the mean measure of $\mu \times \mathfrak{L}((0, 1))$ where $\mathfrak{L}$ is the Lebesgue measure. By restriction to a subset of the augmented dimension $\Pi' \cap (S \times (0, p))$ and subsequent projection back to S, we get the correct distribution of $PP(p\mu)$.

2.1.2 Sampling from Poisson Processes

Given a measure space $(S, \mathcal{A}, \mu)$, there are various methods to simulate $\Pi \sim PP(\mu)$. An understanding of the techniques will be useful for the remainder of this work so they will be briefly described in this section. These techniques are described in Lewis et al. [1978] and more recently, in an overview paper by Vere-Jones [2009]. The approaches will assume that μ is finite. However, if $\mu(S) = \infty$, they are still applicable through a reduction. Let $A_1, A_2, \ldots \in \mathcal{A}$ be a partition of S and by σ-finiteness, $\mu(A_i) < \infty$ so $\mu \cap A_i$ is finite and can be sampled from. Let $\Pi^i \sim PP(\mu \cap A_i)$ by the superposition property, it follows that $\cup \Pi_i \sim PP(\mu)$.

The first approach takes advantage of the fact that when μ is finite, the location of each event does not depend on the number of events that occur.

$$\begin{gathered}\text{Number of events } n \sim \text{Poisson}(\mu(S)) \\ \text{Independent locations } p_i \sim \mu/\mu(S) \text{ for } 1 \leq i \leq n \\ \Pi = \cup_{i=1}^{n} p_i.\end{gathered} \tag{2.1}$$

The second approach uses the fact that the inter-arrival times of points for a Poisson process on the real number line are distributed according to the exponential distribution. To sample from a homogeneous Poisson process on $\mathcal{R}^+$ with the intensity of 1 everywhere

$$z_i \overset{\text{iid}}{\sim} \text{Exponential}\,(1) \tag{2.2}$$

$$p_i = \sum_{j \leq i} z_j \tag{2.3}$$

$$\Pi = \cup_i p_i \sim \text{PP}\,(1)\,.$$

When the objective is to sample from a non-homogeneous process on $\mathcal{R}^+$, it's possible to use the mapping property to map samples from the simple process onto the more complex one. If the process of interest has mean measure μ (still over $\mathcal{R}^+$), the same exponential increments are used but

$$p_i = \left\{ \inf p : \int_{[0,p]} \mu d\mu \geq \sum_{j \leq i} z_j \right\}.$$

In general, this computation may not have a closed form solution and must be calculated numerically, which may be computationally costly. Obviously, for a process over $\mathcal{R}$, the $\mathcal{R}^+$ and $\mathcal{R}^-$ components can be generated the same way.

A technique proposed by Lewis et al. [1978] avoids the need for inverse cdfs by generating a sample from a dominating Poisson process and subsequently thinning that sample. Specifically, to sample from a Poisson process with the rate function $\lambda(x)$, we sample from a Poisson process with the rate function $\lambda^\star(x) \geq \lambda(x)$, with $\lambda^\star$ chosen to be easy to sample from. Then, for each point x, it's kept with probability $\lambda(x)/\lambda^\star(x)$. This approach is closely linked to rejection sampling [Von Neumann, 1951] and, just like rejection sampling, may be very inefficient, accepting very few of the proposed points unless the dominating distribution $\lambda^\star(x)$ is sufficiently close to $\lambda(x)$.

When the process is over a more complex space, $\mathcal{R}^+ \times \mathcal{F}$, simply sample the projection onto $\mathcal{R}^+$ using one of the techniques above and then, use the marking property to generate labels in $\mathcal{F}$.

2.2 Modeling Events with Poisson and Cox Processes.

For many applications, the natural form for observations is events – events that occur at a particular instant in time, have associated features and are non-independent – one event may cause others to occur. As examples, consider people walking through a doorway, packets sent on a computer networks or revisions committed to Wikipedia. Mathematically, we represent each event as the pair $(t, x) \in \mathcal{R}^+ \times \mathcal{F}$ where t corresponds to the time that a particular event occurs and x represents the associated features and takes on values in a feature space $\mathcal{F}$. Clearly, this can be thought of as a point process.

A dataset is then a sequence of observations $(t_i, x_i) \in \mathcal{R}^+ \times \mathcal{F}$ for i from 1 to n. Equivalently, it can be represented as a counting measure $\sum_{i=1}^{n} \delta_{(t_i,x_i)}$. We write D to denote the dataset as a collection of events and will write $(t, x) \in D$ to denote an event in D. We will further use $D_{a:b}$ to mean $D \cap ((a, b] \times \mathcal{F})$, the events that occur between times a and b. While an event-generating process may eventually cause an infinite number of events, any empirically observed dataset must contain a finite number of events and so must end at a certain time T. Note that for a particular observed dataset, it's important to explicitly record and use the beginning and end times of observation and not simply use the timestamps of the first and last events, as the waiting time until the first event is also information that needs to be incorporated.

For a collection of events, we have to explain

1. How many events occur?
2. At what times do the events occur?
3. What features do the events possess?

2.2.1 Homogeneous Poisson Model

There exists a very simple model that answers these questions in the following ways:

1. The number of events is distributed Poisson(α) for some parameter α.
2. An event can occur at any time and no instant is more likely than another, so the times are uniformly distributed on the interval considered.
3. The features of an event do not depend on its time and are drawn independently from a fixed distribution g. Selecting g is a separate problem and there exist well known methods for density estimation.

The second and third answers are clearly chosen for their simplicity and the first can be motivated by the "limit of independent binomials" interpretation of Poisson random variables

and follows from assuming independence between disjoint sets. In mathematical terms, this model corresponds to assuming that

$$\begin{aligned} \text{the density } f_\alpha(t,x) &= \alpha T \mathbf{1}(0 \le t \le T)\, g(x) \\ \text{the data } D &\sim \text{PP}(f_\alpha). \end{aligned}$$

where $g(x)$ is a fixed distribution on the marks $x \in \mathcal{F}$ and α is a scaling parameter. In 1 unit of time, the expected number of events generated by the Poisson process is α and the marks $x_i \overset{\text{iid}}{\sim} g$.

Given an observation of events from time 0 to T, fitting this model entails fitting the only free parameter, α, that determines the expected number of events in any time interval of length one. By standard Poisson results, the MLE is $|D|/T$.

2.2.2 Non-Homogeneous Poisson Model

The model of the previous section assumes that the intensity of the Poisson process is homogeneous over time and that the marking distribution $g(x)$ is fixed and given. Both assumptions are easy to generalize. Let the parameter $\theta = (\alpha, \theta_g, \theta_h)$ and consider models of the form

$$\begin{aligned} D &\sim PP(f_\theta) \\ f_\theta &= T \cdot \alpha \cdot h_{\theta_h}(t) g_{\theta_g}(x) \end{aligned}$$

where

α is the average occurrence rate per unit of time

h_{θ_h} is the temporal distribution for the events with $\int_0^T h_{\theta_h}(t)dt = 1$

g_{θ_g} is the marking distribution with $\int_{\mathcal{F}} g_{\theta_g}(x)dx = 1$.

For example, for h, we can use a function that captures the periodicity of activity due to time-of-day effects or a trivial step function that takes on value $\frac{\theta_h}{T}$ during the day and distributes the remainder of the mass across the night period. For α, the MLE is once again $|D_{0:T|}|/T$. For θ_g, the MLE is the MLE for $x_i \overset{\text{iid}}{\sim} g_{\theta_g}$ and similarly, for θ_h, it is the MLE for $t_i \overset{\text{iid}}{\sim} h_{g_h}$. As long as the separate estimation problems are tractable, no new complexity is introduced.

As an example, consider a model, parameterized by $d_0, d_1, \ldots, d_{23}$ such that $\sum_i d_i = 1$ where d_i is the fraction of events that occurs on the i^{th} hour of each day (and the parameters are shared between days). In this case, $h_{\theta_h}(t) = \theta_{h,\lfloor t \bmod 24 \rfloor}$ and the MLE estimates for the parameters $\theta_{h,i}$ is the fraction of observed events that occurred in that hour.

2.2.3 Cox Process Models and HMMs

The models so far exhibit the standard Poisson process independence structure – the number of events that occur in two non-overlapping time intervals must be independent. However, this independence structure is not realistic for many sources of events. There is a broader range of dynamics that we may want to model and one step in that direction is to switch from Poisson processes to Cox processes.

A Cox process is simply a Poisson process where the underlying mean measure is not fixed but is rather a random variable. When the underlying mean measure is Markov, the object is called a Markov Modulated Poisson Process (MMPP) and is discussed by Rudemo [1972] and Freed and Shepp [1982]. Rydén [1996] describes an EM algorithm for MMPPs. Markov Modulated Poisson Processes have been used as models for packets in a computer network by Fischer and Meier-Hellstern [1993].

Consider a discretization of time into Δ-wide windows and let Z_k denote the value of the hidden state for $\Delta k \leq t < \Delta(k+1)$. Specifying a Hidden Markov Model requires an initial state distribution $P(Z_0)$, a transition distribution $\mathrm{P}(Z_{k+1}|Z_k)$ and an observation distribution $\mathrm{P}(\{t,x\}|Z_k)$. As abuse of notation, let $Z(t) = Z_{\lfloor t/\Delta \rfloor}$, the value of the hidden state at time t.

$$\begin{aligned} f_{\theta,Z}(t,x) &= \alpha_{Z(t)} g_{\theta_{Z(t)}}(x) \\ \{t,x\}|Z &\sim PP(f_{\theta,Z}(t,x)). \end{aligned}$$

Alternatively and equivalently, this can be treated like a standard HMM where at time $k\Delta$, the observations produced are a Poisson process between $k\Delta$ and $(k+1)\Delta$ and the overall event stream is simply a superposition of all the events produced

$$\{t,x\}|Z \sim \cup_k PP\left(\mathbf{1}(\Delta k \leq t < \Delta(k+1))\,\alpha_{Z_k} g_{\theta_{Z_k}}(x)\right).$$

For a specific set of HMM parameters, the forward-backwards algorithm can be used to integrate out Z. The parameters can also be learned with the standard EM algorithm as applied to HMM learning. Furthermore, continuous-time HMM techniques can be applied to avoid discretization.

2.3 Events Causing Other Events

The Poisson process is a very mathematically convenient model for events but the strong independence assumptions built into the Poisson process are not realistic for a large class of data. We now introduce a way for the occurrence of an event to cause more events to occur as a result. This approach is very similar to the ETAS model of Ogata [1988].

Each event (t, x) will cause a Poisson process of other events, with intensity

$$\begin{aligned} k_{(t,x)}(t', x') &= \alpha(x) g_\theta(x'|x) h_\theta(t' - t) \\ \alpha(x) &\text{ is the expected number of events caused by x} \\ h_\theta(t) &\text{ is Delay distribution} \\ g_\theta &\text{ is Label Transition.} \end{aligned} \tag{2.4}$$

Now, we let Π_0 denote the "background" events caused by a baseline Poisson process with mean measure μ_0, and let Π_i denote the events that occur as a result of events in Π_{i-1}. We can write the model as

$$\begin{aligned} D &= \cup_i \Pi_i \\ \Pi_0 &\sim \text{PP}(\mu_0) \\ \Pi_i &\sim \text{PP}\left(\sum_{(t,x) \in \Pi_{i-1}} k_{t,x}(\cdot, \cdot) \right). \end{aligned} \tag{2.5}$$

Alternatively, we can use F_t to denote the history of the process up to time t and write the Poisson intensity as

$$\lambda(t, x|F_t) = \mu_0 + \sum_{(t_i, x_i) \text{ before } t} k_{t_i, x_i}(t, x) \tag{2.6}$$

In order for this definition to be valid and reasonable, some properties must hold for the k distribution. Temporal causality suggests that an event (t, x) can only cause resulting events at a later time, requiring that $h_\theta(t) = 0$ for $t \leq 0$. Note that $h_\theta(0)$ must also be zero, meaning that when we use well-known distributions for h, we restrict them to the domain of $\mathcal{R}^+$. For the temporal setting, the positivity requirement is a very reasonable restriction; however, if events were to take place on a plane, for example, the requirement of the existence of a topological sort under which all possible causes precede all possible effects may be problematic.

Enforcing this temporal ordering:

1. Allows for useful forecasting: knowing events up to a particular time is all that's required to make predictions about the future.

2. Avoids situations where event A causes event B while event B causes event A

Additionally, it is necessary to avoid the case when the average event causes more than one resultant event, leading to an explosion in the number of events. Let $N(x)$ be the expected number of descendant events for an event with features x and note that

$$\begin{aligned} N(x) &= 1 + \int_{\mathbb{R}\times\mathcal{F}} k_{(0,x)}(t', x')d(t', x') \\ &= 1 + \alpha(x) \int_{\mathcal{F}} N(x')g_\theta(x'|x)dx. \end{aligned}$$

It is required that $N(x) < \infty$ for all x.

2.3.1 View as a Random Forest

In the model above, note that each event was either caused by the background Poisson process or a previous event. If we augment the representation to include the cause of each event, the object generated is essentially a random forest, where each event is a node in a tree with a timestamp and features attached. The parent of each event is the event that caused it; if that does not exist, it must be a root node.

For notation, let $\pi(p)$ be the event that caused p, or $\emptyset$ if the parent does not exist. Usually, this parenthood information is not available and must be estimated, which corresponds to estimating the tree structure from an enumeration of the nodes, their topological sort, timestamps and features. This distribution over $\pi(p)$ is precisely the quantity estimated by the z variables in EM, described later.

2.3.2 Sampling

A model is specified by the initial mean measure μ_0 and the function $k_{(t,x)}(t', x')$. It is possible to sample from the prior of this model in two ways. Thinking in terms of the forest representation, one way is to generate the forest through a graph traversal, where we traverse the forest in some particular order and build it up as needed. An alternative is to generate the events in a time-sorted order using the exponential waiting time property of the Poisson distribution.

Using the representation of equation 2.5, first sample Π_0 by the scheme of 2.1. Then, for each event in Π_0, sample the resultant events that it causes to constitute Π_1 and continue this process until some Π_n is empty. Since the number of events produced by the process on a bounded interval is almost-surely finite, the process must terminate. This scheme, however, produces all the events at once, not in a sequential order, and so cannot be applied to an unbounded interval in order to generate elements of the infinite sequence of events. Of course, this can be solved by partitioning Π_0 into finite subsets and sampling them and all their descendants separately.

This approach corresponds to a breadth-first traversal of the corresponding random forest, where the children of each node are sampled as the node is visited. Another traversal order,

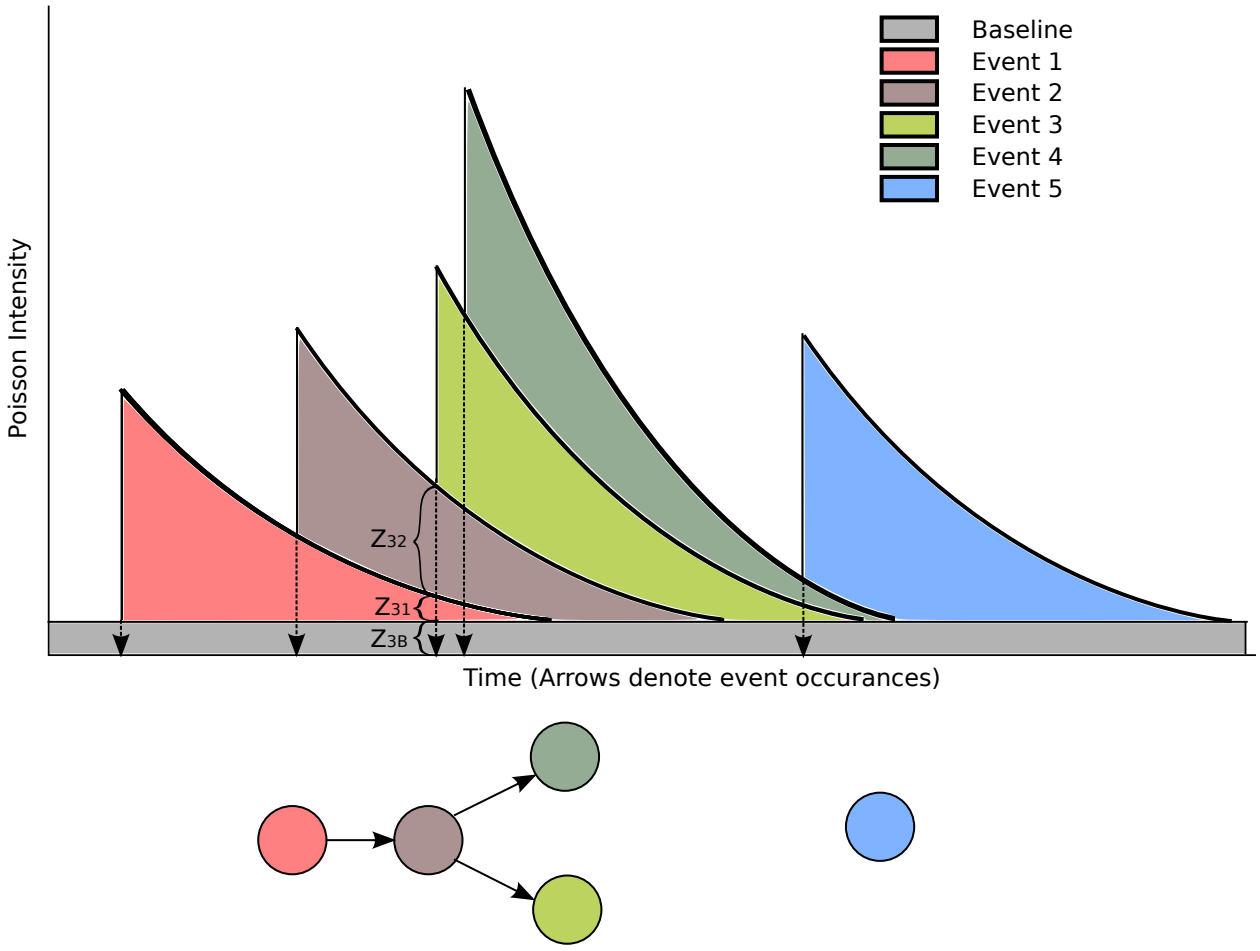


Figure 2.1: A diagram of the overlapping intensities and one possible forest that corresponds to these events.

such as depth first, can be used if more convenient. The key aspect is that traversal order depends on the parent/child relationship of the nodes.

Alternatively, consider sampling the events an order not corresponding to some graph traversal of the random forest, but rather in chronological order. The representation of equation 2.6 combined with the exponential waiting times of the Poisson process allow sequential sampling through the scheme of 2.2. Consider ordering all the events by time from earliest to latest. Knowing $D_{0:t}$ allows you to know the intensity of the process up to the point where a new event is generated, at which point that event itself changes the remainder of the intensity. We use the fact that inter-arrival intervals for points in a Poisson process are

exponentially distributed to sample them

$$\begin{aligned} d_i &\sim \text{Exp}(0) \\ t_{(0)} &= 0 \\ t_{(i+1)} &= \inf\left\{t : \int_{[0,t_{(i)}]\times\mathcal{F}} \left(\mu_0 + \sum_{(t,x)\in D_{0,i}} k\left(t',x'\right)\right) d(t',x') \geq \sum_{j\leq i} d_i\right\} \\ x_{(i+1)} &\sim \text{conditional distribution of x at } t_{(i+1)} \\ D_{0,t} &= \left\{(t_{(i)}, x_{(i)}) : t_{(i)} < t\right\}. \end{aligned}$$

2.3.3 Model Fitting

We can estimate the parameters of the fertility, delay and transition distributions by maximum likelihood. However, the log-likelihood of the model is composed of logarithms-of-sums which cannot be optimized in closed form and pose numerical difficulties to nonlinear optimization algorithms [Veen and Schoenberg, 2008].

The natural solution to this difficulty is an EM algorithm introduced by Dempster et al. [1977] and often used in the machine learning comunity for inference on mixture models. Suppose that $\pi(p)$ the cause of the event, was known for every event. In that case, estimation would be analogous to the estimation problem in 2.2.2. For each Poisson mean measure, there would be multiple samples from the corresponding Poisson process and estimating parameters would essentially involve counting the average number and the delay distributions of the events caused.

However, π, the information on the cause of each event, is latent. In this case, we use EM to iteratively estimate the latent variables and then, maximize the remaining parameters of the model. For uniformity of notation, assume that there is a dummy event $(0, \emptyset)$ and $k_{(0,\emptyset)}(t,x) = f_{base}(t,x)$ so that we can treat the baseline intensity the same as all the other intensities resulting from events:

$$\begin{aligned} \mathcal{L} &= -\lambda + \sum_{(t,x)\in D} \log\left(\sum_{(t',x')\in D_{0:t}v} k_{(t',x')}\left(t,x\right)\right) \\ &\geq -\lambda + \sum_{(t,x)\in D}\left(\sum_{(t',x')\in D_{0:t}} z_{(t',x',t,x)} \log k_{(t',x')}\left(t,x\right)\right) \\ \text{where } \lambda &= \sum_{(t,x)\in D} \int_{[0,T]\times\mathcal{F}} k_{(t,x)}(t',x')d(t',x') \\ \text{subject to} & \sum_{t',x'} z_{(t',x,\prime,x,y)} = 1. \end{aligned}$$

The bound is tight when

$$z_{(t',x',t,x)} = \frac{\log k_{(t',x')}(t,x)}{\sum_{(t',x')} \log k_{(t',x')}(t,x)}.$$

Now, these z variables act as soft-assignment proxies for π and allow us to compute sufficient statistics for estimating the parameters for f_{base} and k. The specifics of how those are computed depend on the specific choices made for f_{base} and k, but this basically reduces the estimation task to that of estimating the distribution from a set of weighted samples.

For example, if $f_{base}(t,x) = \alpha \mathbf{1}(0 \le t \le T_{Max})\, g_f(x)$ where $g_f(x)$ is some labeling distribution,

$$\hat{\alpha}_{MLE} = \frac{\sum_{(t,x)} z_{(0,\emptyset,t,x)}}{T_{Max}}.$$

One implication of the factoring in 2.2.2 is that regardless of delay and labeling distributions and the relative intensities of different events, the total intensity of the total mean measure should be equal to the number of events observed. This can either be treated as a constraint during the M step if possible (for example, if the $\alpha(x)$ has a simple form), or the results of the M step should be projected onto this set of solutions by scaling k and f_{base} by some scalar in order to ensure that the total mean measure integrates to the observed number of events. This step can only increase the likelihood for the new parameters.

2.3.4 Additive Components.

Using the tools above, it is possible to develop more sophisticated models by making $k_{(t,x)}(\cdot,\cdot)$ more complex. In this section, we will consider having

$$k_{(t,x)}(t',x') = \sum_{l=1}^{L} k^{(l)}_{(t,x)}(t',x').$$

where $k^{(l)}$ are individual densities. Mixture probabilities are not required, as the individual components each include a scale term and the total intensity does not need to integrate to any particular sum. What this means is that k is a mixture of multiple densities – for example, $k^{(1)}_{(t,x)}$ can produce events very similar to x at a time close to t, whereas $k^{(2)}_{(t,x)}$ can correspond to more thoughtful responses that occur later but also differ more substantially from the event that caused them.

In any case, this does not substantially complicate inference, only notation. Since the EM algorithm described above introduces a z variable for every additive component inside the logarithm, the separation of some components into a further sum can be handled by introducing individual EM variables for each subcomponent of the sum, estimating the probability

distribution over the subcomponents and computing sufficient statistics as before. The additive terms pass straight through the EM algorithm and now, the credit-assigning step builds a distribution not only over the past events that were potential causes, but also the individual components of the mixture.

2.4 Building the Fertility Model

A key design choice in developing a model is the choice of $\alpha(x)$, the expected number of events. When x takes on a small number of values, it may be possible to directly estimate $\alpha(x)$ for each x, as is done in Chapter 3. However, with a larger feature space, this approach is infeasible for both computational and statistical reasons and so a functional form of the fertility function must be learned. For these fertility models, we assume that x is a binary feature vector.

2.4.1 Linear Fertility

Suppose the label x is a binary feature vector. We can parameterize

$$\begin{aligned} \alpha(x) &= \alpha_0 + \beta^T x \\ \text{subject to} \quad & \alpha_0 \geq 0, \beta \geq 0. \end{aligned}$$

Since $\text{Poisson}(\alpha_0 + \beta^T x) \stackrel{d}{=} \text{Poisson}(\alpha_0) + \sum_{i:x_i=1} \text{Poisson}(\beta_i)$, it's possible to factor $\alpha(x)$ into $\alpha_0 + \sum_{i:x_i=1} \beta_i$ and, as part of EM, build a distribution over which feature generated the event and collect sufficient statistics to estimate the values. Note that $\beta \geq 0$ is an important restriction, since the mean of each of the constituent Poisson random variables must be non-negative. However, this can be somewhat relaxed by rewriting $\beta = \beta^+ - \beta^-$ and considering

$$\begin{aligned} \alpha(x) &= \alpha_0 + \beta^{+T} x + \beta^{-T} (\mathbf{1} - x) \\ &= \left(\alpha_0 - \sum_i \beta_i^- \right) + \beta^{+T} x \\ \text{subject to} \quad & \alpha_0 \geq \sum_i \beta_i^-. \end{aligned}$$

Completely forgoing the $\alpha_0 \geq \sum_i \beta_i^-$ restriction allows the intensity to be negative which does not make probabilistic sense.

2.4.2 Multiplicative Fertility

The linear form of fertility is not always appropriate, as it places significant limits on the negative influence that features are allowed to exhibit and implies that the fertility effect of any feature will always be the same regardless of its context. As an alternative, we can estimate

$$\alpha(x) = \exp\left(\beta^T x\right).$$

where we assume that one of the dimensions of x is a constant 1.

2.4.2.1 Basic Poisson Random Variables

First, consider a simpler setting where we have some observations of Poisson variables and associated binary features x. We wish to model the intensity as the GLM

$$\begin{aligned} n^{(i)} &\overset{\text{ind}}{\sim} \text{Poisson}\left(\exp\left(\beta^T x^{(i)}\right)\right) \\ x^{(i)} &\in \{0,1\}^m. \end{aligned}$$

This is equivalent to

$$\begin{aligned} w_j &= \exp\beta_j \\ n^{(i)} &\overset{\text{ind}}{\sim} \text{Poisson}\left(\prod_{ij} x_j^{(i)} \exp\left(z_j\right)\right) = \text{Poisson}\left(\prod_{ij} w_j^{x_j^{(i)}}\right). \end{aligned}$$

To compute the maximum likelihood estimate of w, it is possible to compute the log-likelihood

$$\begin{aligned} L\left(w\right) &= \sum_i \log\left(\frac{\exp\left(-\prod_j w_j^{x_j^{(i)}}\right)\left(\prod_j w_j^{x_j^{(i)}}\right)^{n^{(i)}}}{n^{(i)}!}\right) \\ &= \sum_i \left(-\prod_j w_j^{x_j^{(i)}} + n^{(i)} \log\left(\prod_j w_j^{x_j^{(i)}}\right) - \log n^{(i)}!\right) \\ \frac{dL}{dw_k} &= \sum_i \left(-x_k^{(i)} \prod_{j\neq k} w_j^{x_j^{(i)}} + \frac{n^{(i)}}{w_j} x_j^{(i)}\right). \end{aligned}$$

This can now be used to optimize for w with coordinate descent

$$w_j \leftarrow \frac{\sum_i n^{(i)} x_j^{(i)}}{\sum_i x_k^{(i)} \prod_{j \neq k} w_j^{x_j^{(i)}}}, \tag{2.7}$$

or by taking steps along the derivative. Note that 2.7 is only a valid update when the rest of the parameters w are held fixed and simultaneously applying that update equation can lead to divergence, so if parameters are jointly updated, it has to be treated as a gradient.

2.4.2.2 In the Poisson Process Model

Now, this same principle holds in the larger model. Assume that x includes a constant term. In the log-likelihood bound formed by EM, the relevant terms are

$$\begin{aligned}
-\sum_{(t,x)\in D} \prod_i (w_i^{x_i}) + \sum_{(t,x)\in D} \left(\sum_{(t',x',t,x)\in D_{0:t}} z_{(t',x',t,x)} \log k_{(t',x')}(t,x) \right) &= \\
-\sum_{(t,x)\in D} \prod_i (w_i^{x_i}) + \sum_{(t,x)\in D} \left(\sum_{(t',x',t,x)\in D_{0:t}} z_{(t',x',t,x)} \sum_i \log w_i^{x_i} \right) + C &= \\
\frac{d}{dw_j} \left(-\sum_{(t,x)\in D} \prod_i (w_i^{x_i}) + \sum_{(t,x)\in D} \left(\sum_{(t',x',t,x)\in D_{0:t}} z_{(t',x',t,x)} \sum_i \log w_i^{x_i} \right) \right) &= \\
-\sum_{(t,x)\in D} x_j \prod_{i\neq j} w_i^{x_i} + \sum_{(t,x)\in D} \sum_{(t',x')\in D_{0:t}} z_{(t',x',t,x)} \frac{x_j}{w_j}.&
\end{aligned}$$

The exact solution for a single w_j is again clear, so this can be optimized by either coordinate descent or gradient steps.

2.4.2.3 Probabilistic, Thinning-Based Version

We can use a different method, based on Poisson thinnings, to estimate the weights. This approach will have the benefit of having a probabilistic interpretation and integrating well into more intricate models. This idea was inspired by the use of thinnings for nonparametric inference of Poisson rate functions in Adams et al. [2008, 2009].

If we take a sample from a Poisson process with mean measure μ and for each point, with probability p, keep it and with probability $(1-p)$ reject it, the resulting object will be a sample from a Poisson process with mean measure $p\mu$. We'll pretend that each event

produces a Poisson process of proto-events with a large mean measure. Then, every feature will act as a thinner and, with probability $w_j \leq 1$, will allow the proto-event to pass. If a proto-event passes the thinning of every filter, it is observed; if even one feature thins it, the proto-event is not seen.

Generative Process It is possible to generate a sample from PP $(g(x)h(t))$ where $g(x) = \alpha \prod w^T x$ with the following approach:

1. Choose $g_d(x) = \alpha$ and note that g_d dominates g but is still integrable.
2. Generate $\Pi \sim PP(g_d(x)h(t))$.
3. For each $p = (t, x) \in \Pi$, sample $b_i^{(p)} \overset{\text{ind}}{\sim} \text{Bernoulli}\,(w_t)$ and include the point in D only if $\prod b_i^{(p)} f_i(x) = 1$.

This corresponds to generating D by repeatedly thinning Π based on each multiplicative term in the density of the mean measure. In this augmented representation, the complete data is composed of Π and the $b_i^{(p)}$.

Conditioning on observations The usual observation contains information on the points that were generated that survived the thinning but loses information on all the points that where generated by the dominating process but were thinned out. By conditioning on D, the total number of events is constrained to be at least $|D|$. Furthermore, the locations of the thinned out events are sampled directly from the dominating distribution. If an event is observed, all of its thinning variables must be 1; if an event is thinned out, at least one of the thinning outcome variables must be 0.

First, a simple fact about Bernoulli variables:

Remark 4. Let $b_i \overset{\text{ind}}{\sim} \text{Bernoulli}(p_i)$. Then, $E(b_i | \prod_i b_i = 0) = 1 - (1 - p_i)\frac{1}{1-\prod_i p_i} = \frac{p_i - \prod_i p_i}{1 - \prod_i p_i}$

Proof.

$$\begin{aligned} E(b_i) &= E(p_i | \prod_i b_i = 0) P(\prod_i b_i = 0) + E(p_i | \prod_i b_i = 1) P(\prod_i b_i = 1) \\ p_i &= E(p_i | \prod_i b_i = 0) \left(1 - \prod_i p_i\right) + 1 \cdot \prod_i p_i \\ E(p_i | \prod_i b_i = 0) &= \frac{p_i - \prod_i p_i}{1 - \prod_i p_i} \end{aligned}$$

□

The following scheme can be used to sample (Π, b) conditioned on D.

$$\begin{aligned} \text{Total dominating measure } \lambda &= \int g_d(x)h(t)d(t,x) \\ \text{Observed events } p_i &= i^{th} \text{ element of D for } 1 \leq i \leq |D| \\ \text{Thinned events } p_i' &\overset{\text{iid}}{\sim} \frac{1}{\lambda} g_d(x)h(t). \end{aligned}$$

Since we observed $|D|$ events that were thinned, the total number of thinned and unthinned events $|\Pi| - |D|$ must be independent from $|D|$. This follows by considering an augmented process $(t$,x,b) and noting that the observed events are a restriction to (t, x) of the events falling in the set satisfying $\prod_j p_j^{f_j(p)}$ and the unobserved events are those that fell in the compliment of that set.

The thinning outcomes for each feature $b_i^{(p)}$ are only considered if that feature is active, $f_i(p) = 1$. If $p \in D$, the event was observed, it was clearly not thinned out and so, $b_i^{(p)} = 1$. On the other hand, if $p \notin D$, we know that at least one of the $b_i^{(p)}$ must have been 0, so by the remark above,

$$E[b_i^{(p)}] = \frac{p_i - \prod_j p_j^{f_j(p)}}{1 - \prod_j p_j^{f_j(p)}}.$$

We can count up all the observed events that were by definition not thinned, estimate the number of thinned out events and for each one of those, compute $E[b_i^{(p)}]$. Then, this is used to update the estimates of p_j, which is then used to estimate the thinned out data in the next iteration.

2.4.3 Combining Linear and Multiplicative Fertilities

Clearly, it's possible to build a fertility model of the form

$$\alpha(x) = \alpha_0^{(0)} + \beta^{(0)T}x + \exp\left(\alpha_0^1 + \beta^{(1)T}x\right) + \exp\left(\alpha_0^2 + \beta^{(2)T}x\right) \ldots$$

by combining the ideas from the two sections, using EM to distribute credit between the constant, $\beta^{(0)T}x$ and the various $\exp\left(\alpha_0^1 + \beta^{(1)T}x\right)$ terms. As this requires fitting a large number of parameters, discretion is advised.

A special case is when x has a particular structure and there is reason to believe that it is composed of groups of variables that interact multiplicatively within the group, but linearly among groups. In that case, each of the multiplicative models can be used on only a subset of variables.

Additionally, using the thinning approach to multiplicative models, it's possible to build a fertility model of the form

$$\alpha(x) = \alpha_0^{(0)} + \beta^{(0)T} x \cdot \exp\left(\alpha_0^1 + \beta^{(1)T} x\right)$$

and so on by using the method from 2.4.1 to additively combine intensities and 2.4.2.3 to scale them, using the intensity being scaled as the dominating measure for the thinning process.

2.5 Computational Efficiency

According to the equations of the previous sections, computing the intensity of the mean measure at a particular time point requires a summation over all previous events. For a specific choice of delay and transition distributions, this can be done explicitly, as described in 2.5.1. More generally however, it is impossible to completely avoid the summation. However, since the delay distributions must necessarily have decaying tails (since the whole distribution must integrate to 1), it's possible to truncate the summation and only take into account events that occurred within T units of time of the time in question. The appropriate choice of T depends on the shape of the delay distribution; it must be large enough to cover almost all of the probability mass of the delay distribution. Note that when estimating the parameters of the delay distribution, no observations greater than T are possible, so if T is too small and truncates relevant events, the mean of the delay distribution will be biased downward.

2.5.1 Exponential Delay Distributions

For certain selections of delay and transition distributions, it is possible to collapse certain statistics together and significantly reduce the amount of bookkeeping required. Consider a setting in which there is a small number of possible labels, that is, $x_i \in \{1 \dots L\}$ and the delay distribution $h(t)$ used is the exponential distribution $h_\lambda(t) = \mathbf{1}(t > 0)\, \lambda \exp(-\lambda x)$. We can use the memorylessness of the exponential distribution to avoid the need to explicitly build a distribution over the possible causes of each event.

First, consider simply computing the log-likelihood of a particular sequence of events. This requires evaluating the Poisson process density at each of the observed points

$$\sum_{(t,x)\in D} \log\left(f_{base}(t,x) + \sum_{(t',x')\in D} \alpha(x')\, \mathbf{1}(t > t')\, \lambda \exp(-\lambda(t-t'))\, k(x',x)\right).$$

Algorithm 2.1 Likelihood

```
LL        = 0              # Accumulated log-likelihood
TS        = vector of 0s   # TS[i] is the last time Intensity was updated
Intensity = vector of 0s   # The exponential's intensity at time TS[i]
for (t, x) ∈ D in chronological order
   LL += log(f_base(t, x)+Intensity[x]· exp(−λ(t − TS[x])))
   for i such that k(x, i) ≠ 0
      Intensity[i] = Intensity[i] · exp(−λ(t − TS[i])) + α(x)k(x, i)λ
      TS[i] = t
# At time t, once the algorithm has processed all events with
# times less than t, the PP intensity for an event of type i is
# Intensity[i] · exp(-λ(t-TS[i]))
```

As the exponential function quickly decays, it's possible to truncate the inner summation and only consider (t', x') that occurred somewhat recently before (t, x). But, with the exponential distribution, we can use the fact that

$$a\mathbf{1}(t > 0)\exp(t) + b\mathbf{1}(t > t')\exp(t - t') = \begin{cases} a\exp(t) & \text{if } \mathbf{1}(t' \geq t > 0) \\ (b + a\exp(-t')) & \text{if } \mathbf{1}(t > t') \\ 0 & \text{otherwise.} \end{cases}$$

Recursively applied, this allows us to keep track of a vector of intensities, one scalar for every possible feature label, instead of summing over possible causes for the event. By processing the events in chronological order, the intensity vector could be updated each time an event was observed. Algorithm 2.1 provides an algorithm for doing this that also takes advantage of the sparsity of $k(x, x')$, if that exists.

To estimate parameters λ, α and k, we need to accumulate sufficient statistics, weighted by their corresponding EM probability weights. The specifics are in Algorithm 2.2. For α and k, the sufficient statistics are simply the counts of events caused and computing them requires keeping track of the total intensity of the Poisson process for events of type j caused by events of type i and then incrementing counters. For λ, this is the time between an event and its cause; fortunately, the weighted average can be tracked without needing to record individual timings as well. Once collected, the parameters can be updated as in Algorithm 2.3 or alternative, regularized estimates can be used. All the ideas presented here can be used if each event type has a different λ, but the ability to keep a single number, instead of a list of candidate causes, depends on the memoryless property of the exponential distribution, so this approach is only usable with a small category of delay distributions such as the exponential or a mixture of exponentials.

While this computational technique works for only a restricted set of models and has computational complexity $O(|D|z)$ where z is the average number of non-zero $h(x, \cdot)$ entries, it

Algorithm 2.2 Likelihood and Sufficient Statistics

```
# Likelihood computation variables
LL        = 0                  # Accumulated log-likelihood
TS        = L x L matrix of 0s # TS[i,j] is the last time Intensity[i,j]
                                 was updated
Intensity = L x L matrix of 0s # Intensity, at time TS[I,J] for I -> J
Delay     = L x L matrix of 0s # Average delay at TS[I,J] for I->J
kSS, λSum, λWeight are accumulators for sufficient statistics
for (t, x) ∈ D in chronological order
   # Compute likelihood
   TotalIntensity = f_base(t, x) + Σ_i Intensity[i,x] · exp(−λ(t − TS[i, x]))
   LL += log(TotalIntensity)
   # Accumulate Statistics
   for i such that k(i, x) ≠ 0
      weight =  TotalIntensity / Intensity[i,x] · exp(−λ(t − TS[i, x]))
      kSS[i,x].increment(weight)
      λSum+=(Delay[i,x] + (t - TS[i,x]))
      λWeight+=weight
   #Update bookkeeping
   for i such that k(x, i) ≠ 0
      CurI  = Intensity[x,i] · exp(−λ(t − TS[x, i]))
      Delay[x,i] = (Delay[x,i]+(t-TS[i])) · curI / (curI+α(x)k(x, i)λ)
      TS[x,i] = t
      Intensity[x,i] = curI + α(x)k(x, i)λ
```

Algorithm 2.3 Update From Sufficient Statistics

```
λ ← λWeight / λSum
for i from 1 to L
   α[i] ← sum(kSS[i, 1:L]) / number of events of type i
   for j from 1 to L
      k[i,j] = kSS[i,j] / sum(kSS[i, 1:L])
```

is much more computationally efficient than the direct method when there is a large number of somewhat closely spaced events. On the other hand, the explicit truncated EM has the complexity of $O(|D|y)$ where y is the average number of events within the truncation window. In order to be accurate, the truncation window must include all the events that still have non-trivial mass in their associated Poisson process, so with dense data, this can be much larger than z.

2.A Poisson MLE

In the model described in 2.2.1 and 2.2.2, the parameters can be simply estimated by maximum likelihood. This appendix will show the equations, largely for the purpose of clarifying notation. The dataset has $|D|$ events and covers t units of time.

The log-likelihood of the data, under the model from 2.2.2 effectively factors into terms modeling the occurrence rate, temporal distribution and marking distribution parameters.

$$\begin{aligned} L_\alpha(D) &= -\int_{[0,T]\times\mathcal{F}} f_\theta(t,x)d(t,x) + \sum_{(t_i,x_i)\in D} \log f_\theta(t_i,x_i) - \log(|D|!) \\ &= -\alpha T + \sum_{(t_i,x_i)\in D} \log\left(T\alpha h_{\theta_h}(t_i) g_{\theta_g}(x_i)\right) - \log(|D|!) \\ &= -\alpha T + |D|\log\alpha + |D|\log T + \sum_{(t_i,x_i)\in D} \log h_{\theta_h}(t_i) + \sum_{(t_i,x_i)\in D} \log g_{\theta_g}(x_i) - \log(|D|!). \end{aligned}$$

With respect to α, this is minimized at $\alpha = |D|/t$. Both g and h completely factor out from the intensity aspect and can be estimated independently, with the approach depending on the specific distribution used.

Chapter 3

Application: Network Dependency Analysis

3.1 Introduction

The research described in this chapter was motivated by the following real life application in the domain of networked distributed systems: in a modern enterprise network of scale, dependencies between hosts and network services are surprisingly complex, typically undocumented, and rarely static. Even though network management and troubleshooting rely on this information, automated discovery and monitoring of these dependencies remains an unsolved problem and statistical techniques can be a useful tool for shedding light on the complex structure of distributed systems. This chapter discusses the probabilistic and statistical aspects of a tool called Constellation for detecting dependencies within a computer network based on passive observation of packets within the network [Barham et al., 2008, Simma et al., 2008].

Constellation takes a black-box approach to locally (at each computer/server in the network) learn explicit dependencies between its services using little more than the timings of packet transmission and reception. The black-box approach is necessary since any more processing of the incoming and outgoing communication packages would imply prohibitive amounts of overhead on the computer/server. The local models of dependency can then be composed to provide a view of the global dependencies. In Constellation, computers on the network cooperate to make this information available to all users in the network.

Constellation and its application to system wide tasks such as characterizing a networking site service and hosts dependencies for name resolution, web browsing, email, printing, reconfiguration planning and end-user diagnosis are described in Barham et al. [2008]. This chapter focuses on the author's contribution – the probabilistic and statistical building blocks of that system: the probabilistic model used in the local learning, the EM algorithm used to fit the parameters of the model, and the statistics of the hypothesis testing used to de-

termine the local dependencies. The model, initially derived as a continuous version of a cascade of noisy ors, and so called *Continuous Time Noisy Or* (CT-NOR), takes as input sequences of input events and output events and their time stamps. It then models the interactions between the input events and output events as Poisson processes whose intensities are modulated by a (parameterized) function taking into account the distance in time between the input and output events. Through this function the domain expert is able to explicitly encode knowledge about the domain.

This chapter is organized as follows: Section 3.2 describes the model. Section 3.3 describes the EM algorithm for fitting the parameters and also considers the relationship to the noisy or gate. The algorithms and framework for applying the model to dependency discovery and change point detection is described in Section 3.4. That section also contains validation experiments with synthetic data. Section 3.5 contains experiments on real data and results. Finally, Section 3.6 has some conclusions and future work.

3.2 The CT-NOR model

The only information used from each packet is its timestamp, the source and destination machines and the protocol over which the communication occurs. Each packet has a timestamp t_i and features x_i, which are the source and destination machines and the protocol. We will use the word **channel** to refer to all events with the same features[1].

Since the events represent packets in a computer network, there exist some physical constraints on what dependencies are possible. When a packet is sent from machine A to B, the only events that may have influenced it are packets sent from some other machine to A. Any event that does not have A as its destination could not physically cause A to send a packet.

This model is built up as a special case of the framework described in Chapter 2. At any one time, the model, called CT-NOR, considers building the model of packets for a single output channel (packets from machine A to B of a fixed protocol) conditioned on all traffic incoming to A. Since every packet is on some channel, the superposition of all the channels defines a comprehensive model of the traffic in the whole network.

Throughout this chapter, our notation will deviate from the rest of the thesis, as the more general notation would make the descriptions more cumbersome and less clear. At each time, we will consider a single machine and will only build a model of a particular output[2] channel conditioned on all the input channels it is allowed to depend on. Since each channel is the output channel of exactly one machine, we can get a model of all the packets by considering

[1] In the domain of computer networks, a channel refers to a unidirectional flow of networked packets. Thus a channel will be identified by the service (e.g., HTTP, LDAP, etc) and the *IP* address of the source or destination. We identified the packets with events as it is only their time stamp that matters.

[2] We refer to packets sent by the machine under consideration as 'output' packets and packets received by the machine consideration as 'input.'

the superposition of the processes produced for all the individual channels. Furthermore, since at any one time, our analysis concentrates on packets received and sent by a single machine, the model estimation can be performed in a distributed fashion, with each computer responsible for tracking its dependencies.

We will denote

o_l	The time of the lth output event
$i_k^{(x)}$	The time of the kth input on channel x
n	denote the number of output events
$n^{(j)}$	the number of input events on channel j
$w^{(j)}$	The average number of output events caused by each input packet on channel j.

Then event k in input channel j *generates* a Poisson process of output events with the base measure $p_k^{(j)}(t) = w^{(j)} h_\theta(t - i_k^{(j)})$. Since at any one time, we're considering only a single output channel, there's no need for a transition distribution as $w^{(j)}$ subsumes it. $h_\theta(t)$ is the distribution of the delay between an input and the output events caused by it, taking as its argument the delay between the time of the output o_l and the time of the input $i_k^{(j)}$. We use the superposition property of the Poisson process to add the individual intensities together and say that $\{o_l\} \sim PP(\sum_j \sum_{k=1}^{n^{(j)}} p_k^{(j)}(t))$ as the probability of the set of n outputs $\{o_l\}$, $1 \leq l \leq n$. The double sum runs over all the channels and over all input events in the channels. Intuitively, and similarly to the noisy or gate in graphical models [Pearl, 1988], the independence between the between input channels translates into a model where the events in the output channel are "caused" by the presence of input events in the input channels (with some uncertainty).

Before concluding this section, we expand a bit on the delay function h_θ as it is an important part of the model. This function provides us with the opportunity of encoding domain knowledge regarding the expected shape of the delay between input and output events. In our experience using CT-NOR to model an enterprise network we used two specific instantiations: a mixture of a narrow uniform and a decaying exponential and a mixture of a uniform and Gaussian. The uniform distribution captures the expert knowledge that a lot of the protocols involve a response within a window of time (we call this co-occurrence). The Gaussian delay distribution extends the intuitions of co-occurrence within a window to also capture dependencies that can be relatively far away in time (such as with the printer). The left tail of the Gaussian corresponding to negative delays is truncated. The exponential distribution captures the intuition that the possibility of dependency decays as the events are further away in time (this is true for the HTTP protocol). We will not explicitly expand these functions in the derivations, as they tend to obscure the exposition.

Groups of channels may have different delay distributions, in which case the delay distribution can be indexed by the channel group and all the derivations in this paper remain the

same. For example, channels can be grouped by network service, where all HTTP channels have the same delay distribution (thus allowing data from multiple channels to assist in parameter fitting), but the DNS channels are allowed a different delay distribution. Additionally, as noted in Chapter 2, we must also include a base measure to generate the initial events that will then trigger some of the other ones observed. Practically, it captures events that are not explained by the remaining channels.

3.3 Estimation and Relationship to Noisy Or.

Estimation can be performed by a single EM algorithm. Let $z_{kl}^{(j)}$ be some positive vector such that $\sum_{jk} z_{kl}^{(j)} = 1$. For a fixed l, $z_{kl}^{(j)}$ is the probability of the latent state indicating that packet k on channel j caused output l. The E-step update is

$$z_{kl}^{(j)} = \frac{w^{(j)} h_\theta(o_l - i_k^{(j)})}{\sum_{j'k'} w^{(j')} h_\theta(o_l - i_{k'}^{(j')})}$$

and in the M-step,

$$\hat{w}^{(j)} = \frac{\sum_{kl} z_{kl}^{(j)}}{n^{(j)}}$$

and

$$\hat{\theta} = \arg\max_\theta \sum_{jkl} z_{kl}^{(j)} \log h_\theta(o_l - i_k^{(j)})$$

For example, for the exponential family, this simply requires moment matching:

$$\mu(\hat{\theta}) = \frac{\sum_{jkl} z_{kl}^{(j)} T(o_l - i_k^{(j)})}{\sum_{jkl} z_{kl}^{(j)}}$$

where $\mu(\hat{\theta})$ is the mean parameterization of the estimated parameter $\hat{\theta}$ and $T(\cdot)$ are the sufficient statistics for the family. All these equations follow trivially from the previous chapter and are provided merely due to the difference in notation.

A predecessor of this model, which explains both the motivation and the CT-NOR name, is a discrete time model based on the very well-known noisy or[Pearl, 1988]. In that case, we bin the observed data into windows of width δ and represent events not by binned counts, but simply by indicators of whether a bin contains any events of a particular type. We say at an event on input channel i in bin t may cause a packet to occur on output channel in bin $t+s$, $s > 0$, with a particular probability $q^{(i)}(s, \delta)$ that depends on the relationship between the

channels and the time difference s. Since multiple input events may cause output packets at time $t+s$ but our observation is just a single indicator, we assume independence and model the probability of an output event in bin $t+s$ as an Or of all the possible caused events.

We continuously shrink the bin size δ by half by selecting a fixed δ_0 and letting $\delta_i = 2^{-i}\delta_0$ while choosing the $q^{(i)}(s,\delta)$ to be consistent, that is, have the probability assigned by the sequence of models to an output event occurring between $\delta_i s$ and $\delta_i\,(s+1)$ converge to some number. In the limit, this leads to the continuous-time, Poisson process based CT-NOR.

3.4 Dependence Discovery and Change Point Detection

With the probabilistic framework described so far, we can use statistical machinery to perform inference for two applications: a) input-output relation discovery and b) change-point detection. The next two subsections describe the algorithms in detail and also validate the main assumptions using synthetically generated data. The final subsection (Subsection 3.4.3) describes a computationally effective approximation to the hypothesis test procedures.

3.4.1 Dependence Discovery

For the purposes of network management, a crucial problem is dependence discovery. For each computer in the network, we are interested in automatically finding out from observations which input channels have a causal effect on an output channel. This information can then be used for scheduling upgrades, troubleshooting problems and identifying bottlenecks.

We can frame the dependency discovery task as hypothesis testing. Specifically, testing whether an input channel j causes output events corresponds to testing the hypothesis that $w^{(j)} = 0$. One way of testing this hypothesis is through the likelihood ratio test [Wasserman, 2004]. We fit two models: M_{full}, under which, all the parameters are unrestricted, and M_{res}, under which $w^{(j)}$ is constrained to be zero. The test statistic in this case is

$$-2\log\Lambda = -2\log\frac{L_{M_{\text{res}}}(\text{Data})}{L_{M_{\text{full}}}(\text{Data})}$$

The asymptotic distribution of this test statistic is called a $\bar{\chi}^2$ and is a mixture of multiple chi-squared distributions with different degrees of freedom. As discussed in Lindsay [1995], the weights depend on the Fisher information matrix and are difficult to compute, but the significant terms in the mixture are χ^2_1 and χ^2_0 which is a delta function at zero. The $\bar{\chi}^2$ emerges as the null distribution instead of the more familiar χ^2 because the weight parameters $w^{(\cdot)}$ are constrained to be non-negative, and when an estimated $\hat{w}^{(j)}$ is zero in the unconstrained model, imposing the constraint does not change the likelihood. If a set of true null hypotheses is known, the mixture coefficients can be trivially estimated, with the weight of χ^2_0 being the proportion of test statistics that are 0. When no ground truth is available,

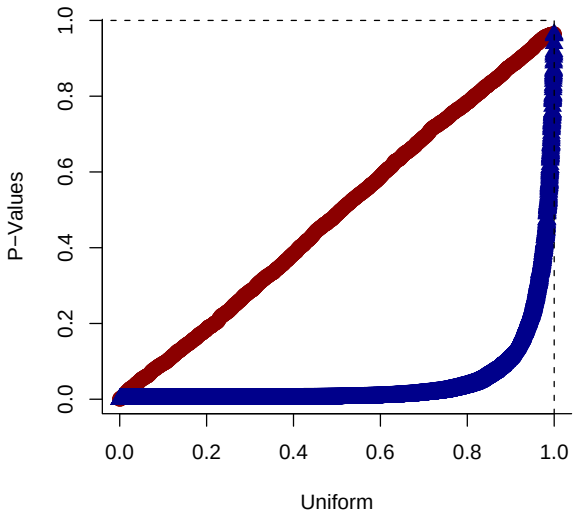


Figure 3.1: Dependency Discovery p-values on Synthetic Data. The red circles (center line) correspond to the cdf of the p-values under the null hypothesis and demonstrate that the p-values are calibrated, since the cdf has a proper uniform distribution. The blue triangles (lower right curve) are the cdf of the p-values under alternative.

the proportion of null hypotheses can be estimated using the method described by Storey [2002] and then used to estimate the mixture proportions.

To demonstrate that the model efficiently recovers the true causal channels and has the proper test-statistic distribution under the null hypothesis, we first test the model on synthetic data that is generated according to some instantiation of the model. 10 input channels are generated; half of them have no causal impact on output events and half produce a Poisson(0.01) number of output events with the delay distribution of Exponential(0.1). Note that the causality is weak – very few input events actively produce an output. For each hour, 500 input events per channel, the corresponding output events, and 100 uniformly random noise events (which are not caused by any input activity) are produced. The resulting p-values are plotted in Figure 3.1.

Observe that the null p-values (conditioned on the test statistic being non-zero) are distributed uniformly. This is evidenced by the p-values following the diagonal on the quantile-quantile plot. The alternative p-values (without any conditioning) for channels that exhibit causality are mostly very low, with 88% being below 0.1. Furthermore, the specific parameter estimates (the delay distribution parameter and $w^{(j)}$) are in line with their true values.

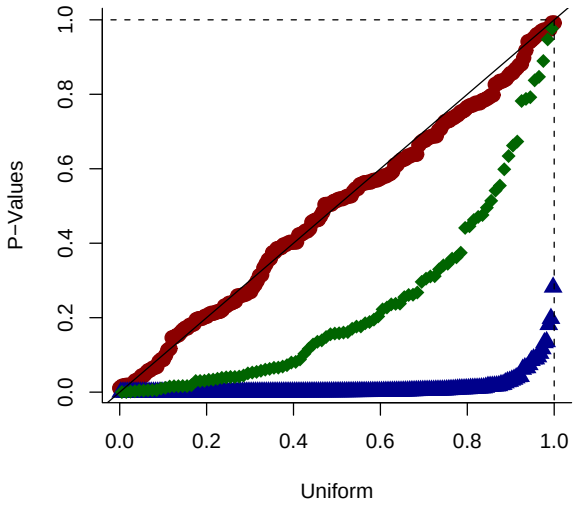


Figure 3.2: P-values for Changepoint Detection on Synthetic Data. The red circles (center line) are the cdf of the p-values under the null where $w^{(j)}$ does not change; they are uniform and so, the p-values are calibrated. The green diamonds(middle curve) is the cdf for p-values on instances with weak changes, where $w^{(j)}$ increases from 0.01 to 0.02) and the blue triangles (bottom right) are a strong alternative where $w^{(j)}$ increases from 0.01 to 0.05.

3.4.2 Changepoint Detection

When the relationship between events is altered, it can be an indication of a significant change in the system; in the case of Constellation, this is of interest to the system administrators. We describe a building block for identifying whether the parameters $w^{(j)}$ change between two time periods and demonstrate its correct functionality. Changepoint algorithms have long been studied in machine learning and statistics, and our test for whether the behavior of a parameter is altered between two time periods can be plugged into one of many existing algorithms. Furthermore, the simple two-period test described here is sufficient for many monitoring applications.

We again use the log-likelihood ratio test methodology. In order to do that, it is necessary to extend the model to allow the parameters to depend on time. The model can be written as

$$\{o\} \sim PP\left(\sum_j \sum_k w^{(j)}_{i^{(j)}_k} h_\theta(o_l - i^{(j)}_k)\right).$$

Detecting changepoints is accomplished by testing two hypotheses. The null is that the weights do not change between two time periods, and can be written as $w^{(j)}_t = w^{(j)}$. Under

the alternative, for a particular channel of interest m and an interval of time S, the weight changes:

$$\begin{aligned} \forall j \neq m \quad w_t^{(j)} &= w^{(j)} \\ w_t^{(m)} &= w^{(m)} \text{ if } t \in S, w'^{(m)} \text{otherwise.} \end{aligned}$$

The existence of a changepoint is equivalent to rejecting the null hypothesis. Fitting the alternative model is a simple modification of the EM procedure described for the null model; for fast performance, it is possible to initialize at the null model's parameter values and take a single M step, reusing the latent variable distribution estimated in the E step. The test statistic in this case will again be $-2 \log \Lambda$ and its null distribution will be χ^2 if the true $w^{(m)} > 0$ and $\bar{\chi}^2$ otherwise.

Figure 3.2 shows a quantile-quantile plot of the p-values (computed using the χ^2 distribution) under the null hypothesis, computed for causal channels of the same synthetic data as in section 3.4.1; there are two hours of data with 500 input events per channel per hour. As expected, the quantile-quantile plot forms a straight line, demonstrating that on the synthetic dataset, the null test statistic has a χ^2 distribution. When a strong changepoint is observed ($w^{(j)}$ changes from 0.01 to 0.05) , the p-values are very low. When a weak changepoint is observed ($w^{(j)}$ changes from 0.01 to 0.02) the p-values are lower than under the null distribution but power is significantly lower than when detecting the major changepoint.

3.4.3 Efficiency — Bounding the Log-Likelihood Ratio

Computing the log-likelihood ratio requires refitting a restricted model, though only a small number of EM steps is typically required. The process can be computationally expensive, as the number of tests required grows linearly with the number of machines in the network. However, it is possible to bound the log likelihood ratio for dependency discovery very efficiently.

For the restricted model testing channel m's causality, we must compute the likelihood under the constraint that $w^{(m)} = 0$. Take the estimates of w of the unrestricted model and let $\alpha = \frac{\lambda}{\lambda - w^{(m)} n^{(m)}}$. Instead of computing the ratio with the true maximum likelihood parameters for the restricted model, we propose a set of restricted parameters, and compute the ratio using them. We produce a restricted version of parameters $w^{(\cdot)}$ by setting $w^{(m)}$ to zero and inflating the rest by a factor of α. That simply corresponds to imposing the restriction, and redistributing the weight among the rest of the parameters, so that the expected number of output packets remains the same. In that case,

$$
\begin{aligned}
-2\log\Lambda &= -2\log\frac{L_{M_{res}}(Data)}{L_{M_{full}}(Data)} \\
&\geq -2\log\prod_l \frac{\sum_{j\neq m,k}\alpha w^{(j)}h_\theta(o_l - i_k^{(j)})}{\sum_{jk} w^{(j)}h_\theta(o_l - i_k^{(j)})} \\
&= -2\log\prod_l \alpha\left(1 - \frac{\sum_k w^{(m)}h_\theta(o_l - i_k^{(m)})}{\sum_{jk} w^{(j)}h_\theta(o_l - i_k^{(j)})}\right) \\
&= -2\log\prod_l \alpha\left(1 - \sum_k z_{ml}^{(j)}\right).
\end{aligned}
$$

As a reminder, z_{ml}^{j} is the latent variable distribution estimated in the E-step of EM. Since the numerator of the log-likelihood ratio is a lower bound and the denominator exact, this expression is a lower bound on Λ. Intuitively, $\log\prod_l\left(1-\sum_k z_{ml}^{(j)}\right)$ corresponds to the probability that channel m has exactly 0 output events assigned to it when causality is assigned according to the EM distribution on the latent variables. The $\log\alpha$ term corresponds to the increase in likelihood from redistributing channel m's weight among the other channels.

This approximation should work well when the removal of a channel causes only minor relative changes in the remaining parameters of the models. On the other hand, it's expected that the bound will be loose when it is possible to shift many of the output packets produced by the channel in question onto a different channel, as the bound will only consider the first-order effects and not the implication of that channel's fertility increasing. However, as the results will show, the bound works very well for the domain described in this section.

3.5 Results

We describe the results of applying the algorithms of the previous section to a subset of a real dataset consisting of a trace comprising headers and partial payload of around 13 billions packets collected over a 3.5 week period in 2005 at Microsoft Research in Cambridge, England. This site contains about 500 networked machines and the trace captures conversations over 2800 off-site IP addresses. Ground-truth for dependence discovery and change point detection is not readily available and it has to be manually generated. We took 24 hours of data at the web proxy and manually extracted ground truth for the HTTP traffic at this server by deep inspection of HTTP packets. It is with this part of the data that we validate our algorithms, as it provides us with objective metrics, such as precision and recall, to assess the performance of our algorithms.

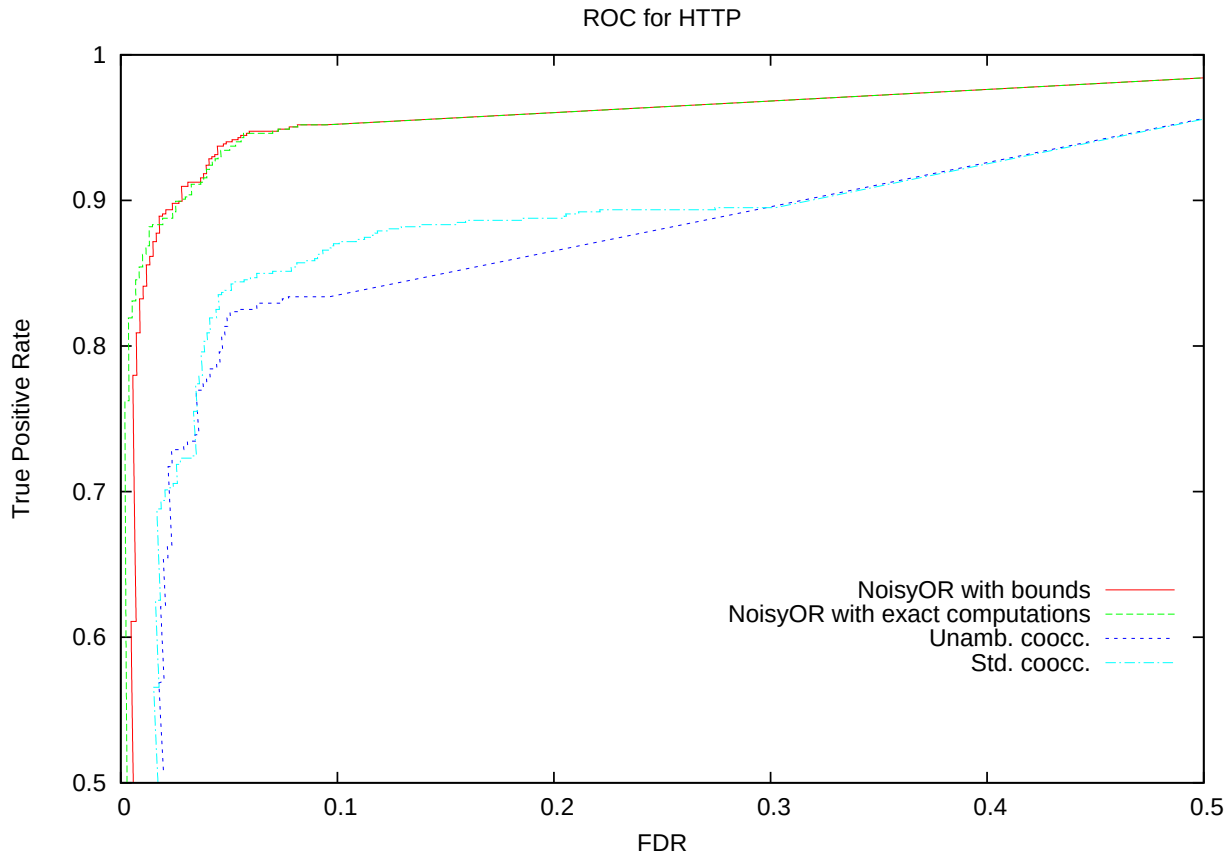


Figure 3.3: ROC Curve for Dependency Discovery on Labeled HTTP

3.5.1 Dependency Discovery

First, we are interested in assessing the performance of the dependence discovery capabilities of our model and hypothesis testing algorithm. In the application of diagnosis and monitoring of networked systems it is crucial to maintain a consistent map of all the server and services inter-dependencies and their changes. Finding dependencies at the server level is the main building block used by Constellation [Barham et al., 2008] in building this global map. We compare our method to two other alternatives. One is a simple binomial test: for each input channel, we count the number of output packets falling within a W width window of an input packet, and determine whether that number is significantly higher than if the output packets were uniformly distributed. We call this "standard co-occurrence." The second alternative considers an input and output channel to be dependent only if there is a unique input packet in the immediate vicinity of an output packet. The reason we select these two alternatives is that a) they reflect (by and large) current heuristics used in the systems community by work like Bahl et al. [2007] and b) they will capture essentially the "easy" dependencies (as

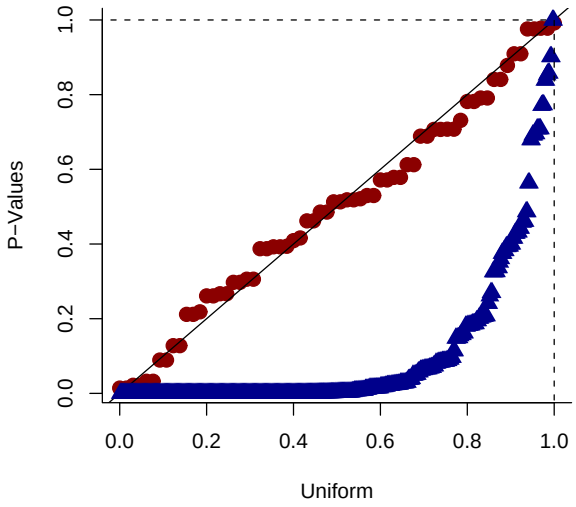


Figure 3.4: P-values for Changepoint Detection. Again, the red circles are the cdf of the p-values on hypotheses that are null according to the ground truth and their proximity to the center diagonal demonstrate the proper calibration. The blue triangles on the lower-right are the cdf of the p-values for pairs in which there is a dependency according to the ground truth.

our results indicate).[3] As can be seen on the ROC curve in Figure 3.3, CT-NOR successfully captures 85% of the true correlations with a 1% false positive rate. In total, the model detects 95% of the true correlations at 10% of false positives. We want to additionally point out that some of correlations present are very subtle; 13% of the correlations are evidenced by a single output packet. We also point out that CT-NOR performs significantly better than both alternatives based on co-occurrence of input packets, providing more evidence that CT-NOR is capturing nontrivial dependencies. The approximation error from using the bound of section 3.4.3 is minimal, while the computation savings are significant. The bounds on log-likelihood ratio test for a hour of traffic on a busy HTTP proxy can be computed in 7 seconds; exact computations take 86 seconds.

3.5.2 Changepoint Detection

Since the true presence or absence of a changepoint is unknown, we estimate it from the actual packet causes, obtained through deep inspection of HTTP packets. We collect a set

[3] As sometimes an input package generates more than one output packet, we enabled our model to account for this by allowing "autocorrelations" to take place. Namely a packet in an output channel can depend on an input channel or on the (time-wise) preceding output packet.

of input and output channel pairs for which there is no evidence of change. We regard these as coming from the null hypothesis. A set of pairs for which the ground truth provides strong evidence of a change are collected, and considered to be from the alternative hypothesis.

We apply our changepoint test to that population, and report the results in Figure 3.4. The CT-NOR changepoint detection algorithm produces uniformly distributed p-values for channels that come from the null hypothesis and do not exhibit a changepoint, confirming that our null hypothesis distribution is calibrated. On the other hand, the test on alternative hypothesis channels produces a large proportion of very small p-values, indicating confidence that a changepoint occurred.

3.6 Discussion

For the particular application of dependency discovery between channels in a computer network, a variety of approaches have been tried. They all failed miserably. One approach is to cast the problem as one of classification. This avenue was tried by collaborators at Microsoft, who discretized time into suitable periods, generated features defined by the existence or absence of events in the input channels and use various classification methodologies to predict the existence or absence of events in the output channel. However, the accuracy was not acceptable. Once we started to look at Poisson as the appropriate way to quantify the distributions in these classifiers the choice of the Poisson process cascades became clear. Another simple approach, referred to as co-occurrence involves counting the number of times that a packet on a particular output channel is preceded by only packets on a single input channel, but upon empirical evaluation, the Poisson-process model produces markedly better performance. Yet another method involves hypothesis testing, comparing the inter-time between events in the input and output channels to the inter-time between the input and a fictitious random channel. The accuracy in terms of false positives and true positives was worse than those based on co-occurrence. The main problem here is that we are considering pairwise interactions, ignoring any "explaining away" that occurs.

We presented a generative model based on Poisson processes called CT-NOR, to model the relationship between events based on the time of their occurrences. The model is induced from data only containing information about the time stamps for the events. This capability is crucial in the domain of networked systems as collecting any other type of information would entail prohibitive amounts of overhead. Specific domain knowledge about the expected shape of the distribution of the time delay between events can be incorporated to the model using a parameterized function. The combination of knowledge engineering and learning from data is clearly exemplified in the application we presented to the domain of computer systems.

This model provides building blocks for diagnosis and monitoring, with algorithms based on statistical hypothesis testing for (a) discovering the dependencies between input and output

channels in computer networks, and for (b) finding changes in expected behavior (change-point detection). We validated these algorithms first on synthetic data, and then on a subset (HTTP traffic) of a trace of real data from events in a corporate communication network containing 500 computers and servers. While the amount of data handled in this section did not pose computational problems even on a single machine, the approach is easy and natural to distribute, with each computer building a model of its own output channels.

Chapter 4

Application: Twitter Messages

Twitter is a popular microblogging website that is used to quickly post short comments for the world to see. Each comment (called a tweet) contains the username of the sender, a timestamp and the body text, which is limited to 140 characters. The comment body is largely free-form text, though several annotation conventions have emerged. This chapter will apply the ideas from Chapter 2 to build a probabilistic model for which messages can be expected to arrive at a particular time, as a function of previous activity.

One of the recently emerging popular uses of Twitter is commentary on the stock markets. Conveniently, the posts reporting news, commentary or opinions on specific stocks enumerate the stocks under discussion with the $ticker tags[1] in the body of the post, allowing users to easily search for new tweets of interest. These messages form a conversation – some are written as explicit responses to previous ones (occasionally denoted by the @username tags). Others are posted not as a direct result of a past Twitter message, but as a result of a real-world event inspiring the commentary.

Figure 4.1 plots the empirical intensity of tweets containing a select subset of tickers. Notice that when looking at aggregate intensities over large time periods, there are obvious daily and weekly periodic components, but the message intensity of some days is much greater than would otherwise be expected. On a smaller time scale, the message frequency is relatively bursty, as certain messages and responses cause moments of intense activity followed by relative silence. We will show that our approach enables us to effectively build a model of the timing and features of the messages.

4.1 Stock Twitter Messages

For this study, we used the Twitter Search API to collect messages containing stock tickers in their body. As the API only returns tweets no older than 8 days, ongoing data collection

[1] The character sequence $XXXX does not always correspond to a stock ticker as the character '$' is often used in place of S, as in '$o'

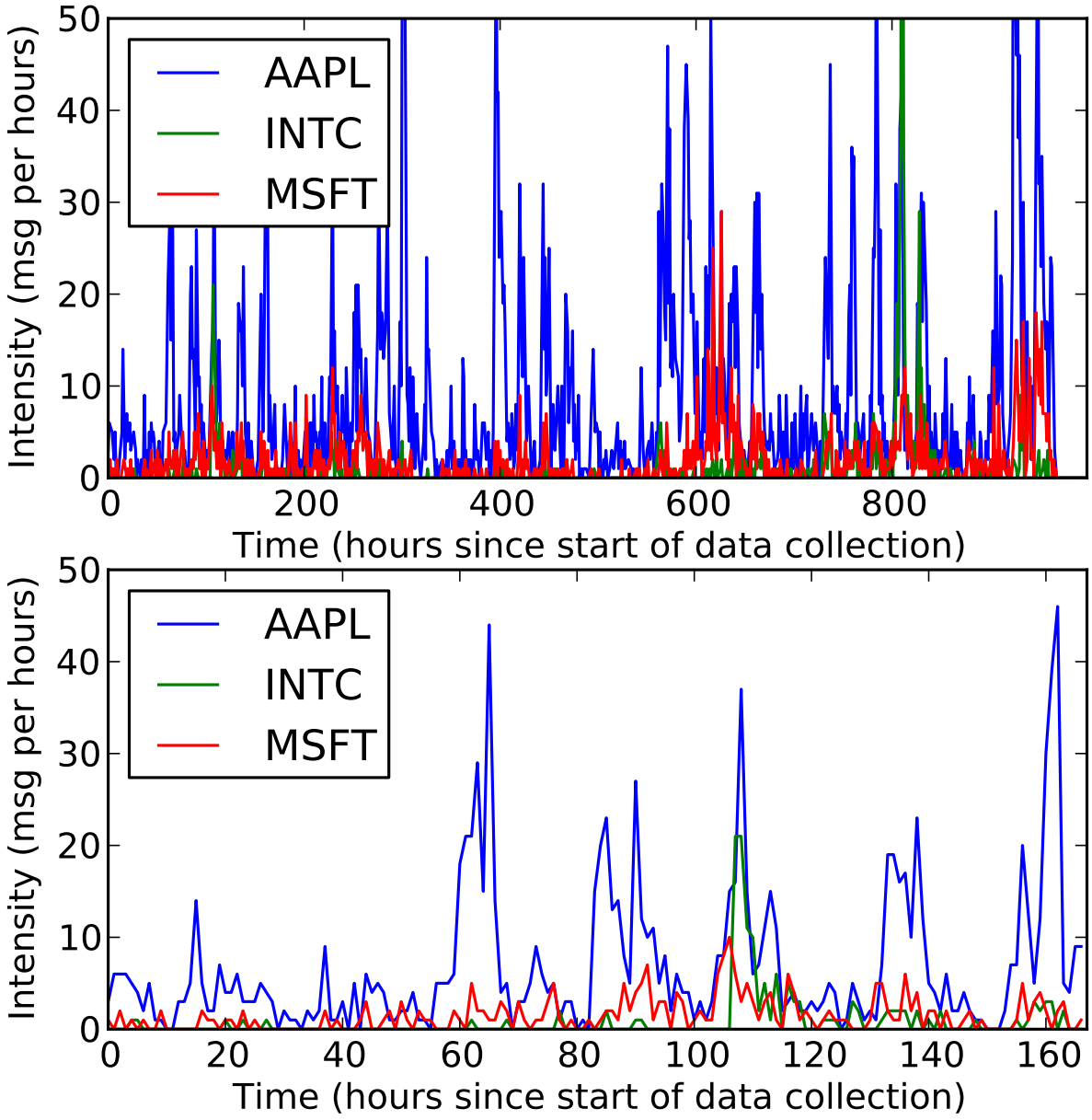


Figure 4.1: Frequencies of tweets tagged with selected tickers. The top plot depicts intensities over the complete data collection period, while the bottom plot is zoomed in on the first week. Note the periodicity.

was required to gather a longer-term dataset. The collected dataset contains 54717 messages and covers a period of 39 days. We will describe a series of models of increasing complexity for the messages and demonstrate the usefulness of the more complex aspects of the model in capturing statistical properties of the data.

For analysis, we split the data into a training (first 34 days) and test(last 5 days) datasets. While the underlying process is probably not stationary and so the testing data is drawn from a different distribution than the training set, this is necessary for a realistic validation since one key application is predicting the properties of future events. Furthermore, if the dependencies between events are largely stationary, the model described here suffers less from non-stationarity of the background process than one without self-excitation.

Each message is represented in terms of the tickers present in the message's body. For the purpose of modeling, each message can be represented as a triple of a user, timestamp and a binary vector of features. For example, a typical message

```
User:  SchwartzNow

Time:  2009-12-17T19:20:15

Body:  also for tommorow expect high volume options traded stocks
like $aapl,$goog graviate around the strikes due to the delta
hedging
```

occurs on 2009-12-17 at 19:20:15 and has the features $AAPL and $GOOG and is missing features such as $MSFT. Another message, with the body

```
$dell has a store?  RT @aiki14 Here's why $AAPL will always win,
you don't see this at the Dell Store http://stk.ly/4vFp0P
```

has features $AAPL, $DELL, HAS_LINK and is posted as a response to a previous message by user aiki14. Note that due to the character limit constraints (as well as Internet culture), the messages tend to not be completely grammatical English and often, a message is simply a wrapper around a link to some Web resource with further information on the topic discussed. In addition to the stocks involved and whether links are involved, features also denote the presence or absence of some key words like 'buy' or 'option.'

4.2 Baseline Intensities

Fundamentally, our models have to describe three aspects of the data – the number of events, the times at which the events occur and the specific features of those events. For Twitter messages, the features of interest are those that describe what stocks the tweets are about. While the aspects are not independent, we will treat them as such for the purpose of building

up a reasonable baseline that we can then elaborate on. We will look at the count and timing aspect first.

The simplest possible baseline intensity is a time-homogeneous Poisson process. As can be seen in Figure 4.1, the number of messages in a particular instant is a periodic function, with periods of high activity falling during times at which the market is open and users are actively commenting. If we disregard the possibility of an event causing other events and simply model the events as coming from a Poisson process with a particular mean measure, the uniform mean measure is an unreasonable one to use. For many domains, a reasonable choice for a baseline intensity is a sinusoid, for example $h(t) = \alpha + \beta \sin(\pi t/12 - t_0)$ where both $\alpha \geq \beta \geq 0$ to ensure that the intensity is non-negative everywhere. However, in this case, the daily intensity is not strictly a sinusoid but varies depending on proximity to the market open or close. We partition the day into intervals assume that the intensity is uniform within the hour and that the pattern repeats. For the purpose of the reported experiments, the intervals are an hour long, so if t is in days, $h(t) = p_{\lfloor t/24 \rfloor}$. The log-likelihoods for just the baselines are reported in Table 4.2 on page 49 but it's worth noting that the gain from using a baseline measure that captures periodicity is much smalled than the gain from the other parts of the model.

This timing model must be combined with a feature distribution. Here, for the baseline, we simply use a fully independent model, where each feature is present independently of each other. That is, $g(x) = \prod_i p_i^{g_i(x)} (1 - p_i)^{1-g_i(x)}$ where g_i is the i^{th} feature. Clearly, the MLE estimates for p_i is simply the empirical fraction of the data that contains that feature.

4.3 Delay Distributions

When events are allowed to cause other events, each event induces a Poisson Process of successor events. We factor the intensity for that process as

$$k_{(t,x)}(t', x') = \alpha(x) g_\theta(x'|x) h_\theta(t' - t)$$

with the constituents described in equation (2.4). When deciding on the intensity for that process, the model needs to account for the expected number of events, their timings and their attributes.

When there is a small set of the types of events, as in the work described in Chapter 3, it is possible for each event type to be associated with a scalar intensity, but in case of the featurized representation, the intensity must be a generalizing function of the features. This can be built up from components that are either linear or multiplicative in the features.

For the delay distribution h, the density used must capture the empirical fact that responses by and large tend to occur a short time after the original message, but at the same time, there exist some responses take quite a long time. So while a large proportion of the probability mass must be close to zero, there needs to be sufficient mass in the tails of the distribution as well. As candidates, we consider the following:

Type	Train	Test
Exponential	-145576.09	-57329.52
Gamma(0.9)	-145363.86	-57257.57
Gamma(0.8)	-145185.88	-57201.27
Gamma(0.7)	-145063.14	-57169.22
Gamma(0.6)	-145028.75	-57174.93
Gamma(0.5)	-145136.13	-57240.44
Unif(0,1000)	-148468.23	-58084.74
Unif(0,2000)	-147371.03	-58083.98
Mix 2 Unif	-146337.19	-57538.84
Mix 4 Unif	-145915.92	-57360.63

Table 4.1: Log-likelihoods for various delay functions.

1. Exponential distribution: $h_\lambda(t) = \lambda \exp(-\lambda t)$
2. Gamma: $h(t) = t^{k-1} \frac{\exp(-t/\theta)}{\Gamma(k)\theta^k}$
3. Mixture of Exponentials: $h(t) = \sum_i p_i \lambda_i \exp(-\lambda_i t)$ with $\sum p_i = 1$, $p_i \geq 0$
4. Uniform: $h(t) = \frac{1}{Z}\mathbf{1}(0 < t \leq Z)$ for a fixed Z
5. Piecewise uniform: $h(t) \propto \sum_i p_i \mathbf{1}(Z_i < t \leq Z_{i+1})$ for bins Z_i.

Log-likelihoods are reported in Table 4.1 on page 43, where the transition function used is h_γ, as described in the next section. Parameters of the distribution are fit with maximum likelihood. For the Gamma delay, we fit both the shape and scale parameters. Additionally, we report results for several fixed shape parameters, for the sake of observing the effect of the shape on performance.

For this task, as well as the Wikipedia modeling described in the next chapter, the delay distribution needs to be effective at capturing the fact that while in most cases, triggered events occur relatively soon after the initial one, it's possible and not completely unlikely for a response to occur after a significant delay. When the delay used is an exponential distribution, where the mean and variance are deterministically linked by the parameterization, the fitted distribution tends to put too little mass away from the mean. The delay distributions that lead to the highest log-likelihood, both on training and test sets, are the higher-variance Gammas with the shape parameter less than 1.

4.4 Transition Distribution

The third remaining aspect of the model is the transition distribution $g(x, x')$ which that specifies the types of events that are expected to result from an event of type x. Lets consider

some ways in which a message can cause other messages to be sent:

1. A simple "retweet" – a duplication of the original message. Empirically, these occur with a significant frequency as individuals repeat messages they believe are interesting so that the individual's followers are also notified.

2. As a response – the message can prompt either a specific response to the content of the message, or simply motivate another message on a similar topic.

3. Another type of message attachment captured by the model, somewhat as an artifact of the model formulation but useful nonetheless. In a period shortly after a sent message, the probability of another, even completely unrelated message is increased because the original event acts as a proxy for general user activity. These kinds of messages can be considered to be unrelated to the "cause" in content, and so should take on a distribution from the background process and represent a residual variation in the baseline event rate that the baseline measure fails to capture.

Consider a transition function parametrized by γ that is a product of independent per-feature transition functions $g_\gamma^{(i)}(f, f')$, each of which is a mixture of the identity function and the feature distribution of the background process. Note that g_γ is *not* a mixture of the identity and the background feature distribution.

$$g_\gamma(x, x') = \prod_i \left((1-\gamma)\, \mathbf{1}\left(x_i = x_i^{'}\right) + \gamma p_i^{x_i'} \left(1 - p_i^{1-x_i'}\right)\right)$$

One obvious special cases are $\gamma = 1$, which means that each resultant event is drawn straight from the background process and its features have no relationship to the event that caused it. Another special case is $\gamma = 0$, where the caused events must be identical to the cause. Figure 4.2 on page 45 shows that both of these extremes yield poor training and testing likelihoods. Note that as γ is increased, the estimated mean decreases. When γ is high, events tend to be associated with the most recent previous event without regard to the features, whereas when γ decreases, the model is willing to look further back to find a feature-compatible cause for an event.

Lets consider the case of $\gamma = 1$ again. With an exponential delay distribution the interpretation of Section 2.5 and $\alpha(x)$ fixed at 1, this is equivalent to stating that the expected number of events is an exponential moving average of the number of recent events, with decay parameter determined by λ. The aforementioned EM algorithm can be used to find the optimal decay parameter, but as the log-likelihood plots show, this EMA model is inferior to one that utilizes the features of the events.

As the results show, the choice of a delay distribution has a smaller impact on the overall likelihood than the transition distribution. This is partially the case because for an individual event, the features take place in a large space and there's more to explain. The predictive ability of an event's Poisson process to explain the specific features of a resultant event is the predominant benefit of the model.

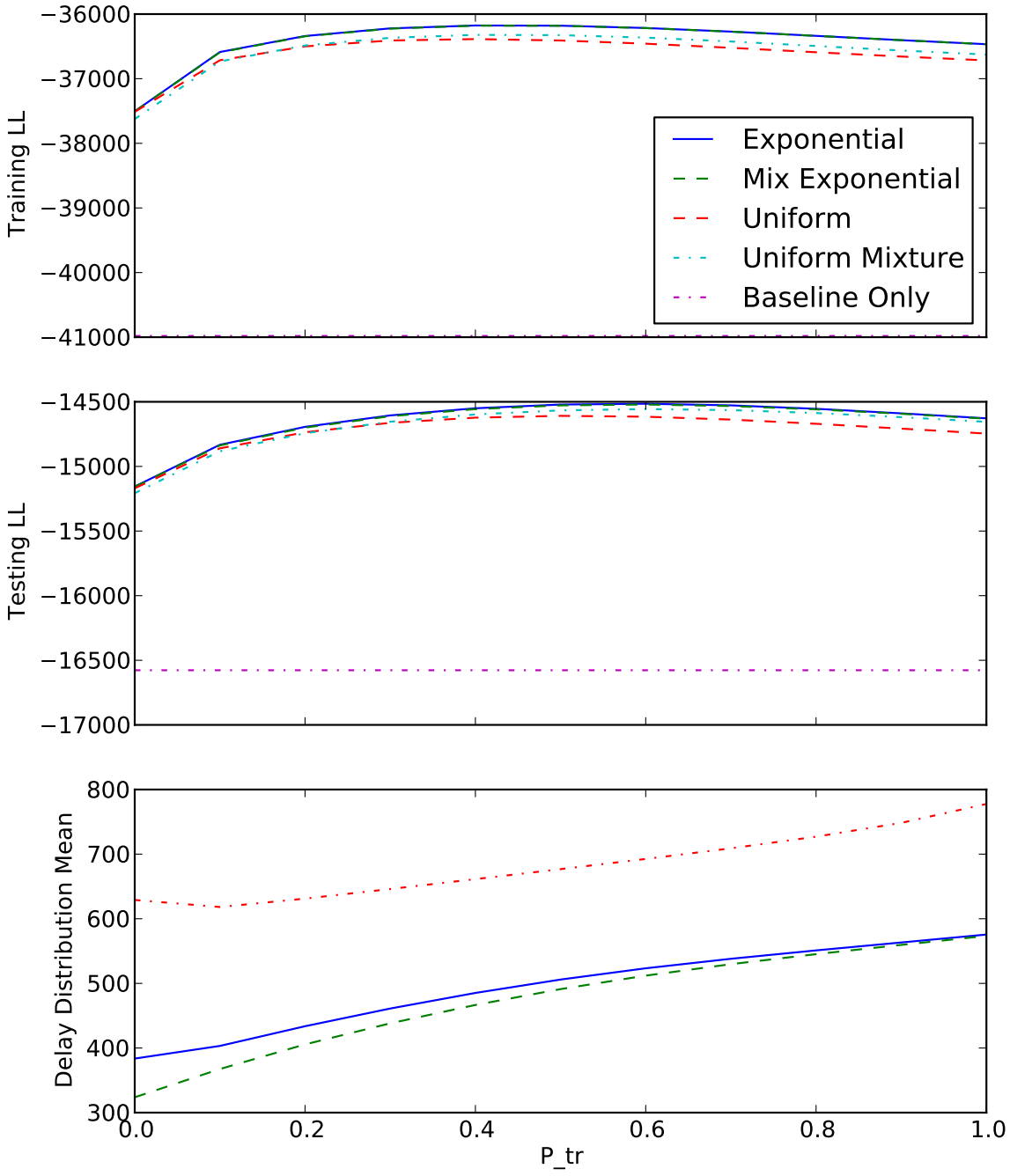


Figure 4.2: Effects of $p_{tr} = 1 - \gamma$ on the log-likelihood and delay distribution mean.

4.5 Expanding the Model

The first paragraph of Section 4.4 enumerates different effects that need to be captured by the model. Different phenomena correspond to different g. For example, the retweet messages are expected to be completely identical to the original, with the possible exception of a "@username" reference tag. A response would have similar features but may differ in few, and a density-proxy message would have features independent of the causing message.

As before, let

$$g_\gamma(x, x') = \prod_i \left((1-\gamma)\, \mathbf{1}\left(x_i = x_i^{'}\right) + \gamma p_i^{x_i'} \left(1 - p_i^{1-x_i'}\right)\right)$$

and note that for some special cases,

$$\begin{aligned} g_0(x, x') &= \prod_i p_i^{x_i'} \left(1 - p_i^{1-x_i'}\right) \\ g_1(x, x') &= \mathbf{1}\left(x = x'\right). \end{aligned}$$

g_1 captures the 'retweet' phenomena while h_0 captures the density-proxy effect. g_γ for $0 < \gamma < 1$ prefers features that look similar to the cause, but allows some features to vary.

Consider a few models, where all the Greek letters represent parameters to be estimated:

$$\begin{aligned} k_{1(t,x)}(t', x') &= \exp\left(\alpha_1 + \beta_1^T x\right) h_1\left(t' - t\right) g_1(x, x') \\ k_{2(t,x)}(t', x') &= \exp\left(\alpha_2 + \beta_2^T x\right) h_2\left(t' - t\right) g_\gamma(x, x') \\ k_{3(t,x)}(t', x') &= \exp\left(\alpha_3 + \beta_3^T x\right) h_3\left(t' - t\right) g(x, x') \\ k_{4(t,x)}(t', x') &= \exp\left(\alpha_4 + \beta_4^T x\right) h_4\left(t' - t\right) \left(\eta_1 g_1(x, x') + \eta_2 g_\gamma(x, x') + \eta_3 g_0(x, x')\right) \\ k_{5(t,x)}(t', x') &= \sum_{i=1}^{3} f_i(t, x, t', x'). \end{aligned}$$

f_i for i from 1 to 3 is designed to capture the i^{th} phenomenon, and f_4 and f_5 jointly capture all the effects. Note that both g and h are distributions, so it's easy to compute $\int f_i(t, x, t', x') dt' dx'$. The results are shown in in Figure 4.3 on page 47, though the result for model 3 is not illustrated as it's far below the range of the axes. Observe that models 4 and 5 are significantly superior to the first three, demonstrating that separating the multiple phenomena is useful. For g, we use an exponential distribution.

The difference between models 4 and 5 is that in model 4, all the transition distributions share the same fertility and delay functions, whereas in 5, each has its own fertility and delay. The fact that the latter model performs significantly better is a sign that the three different categories of message relationships have different associated fertility parameterizations and

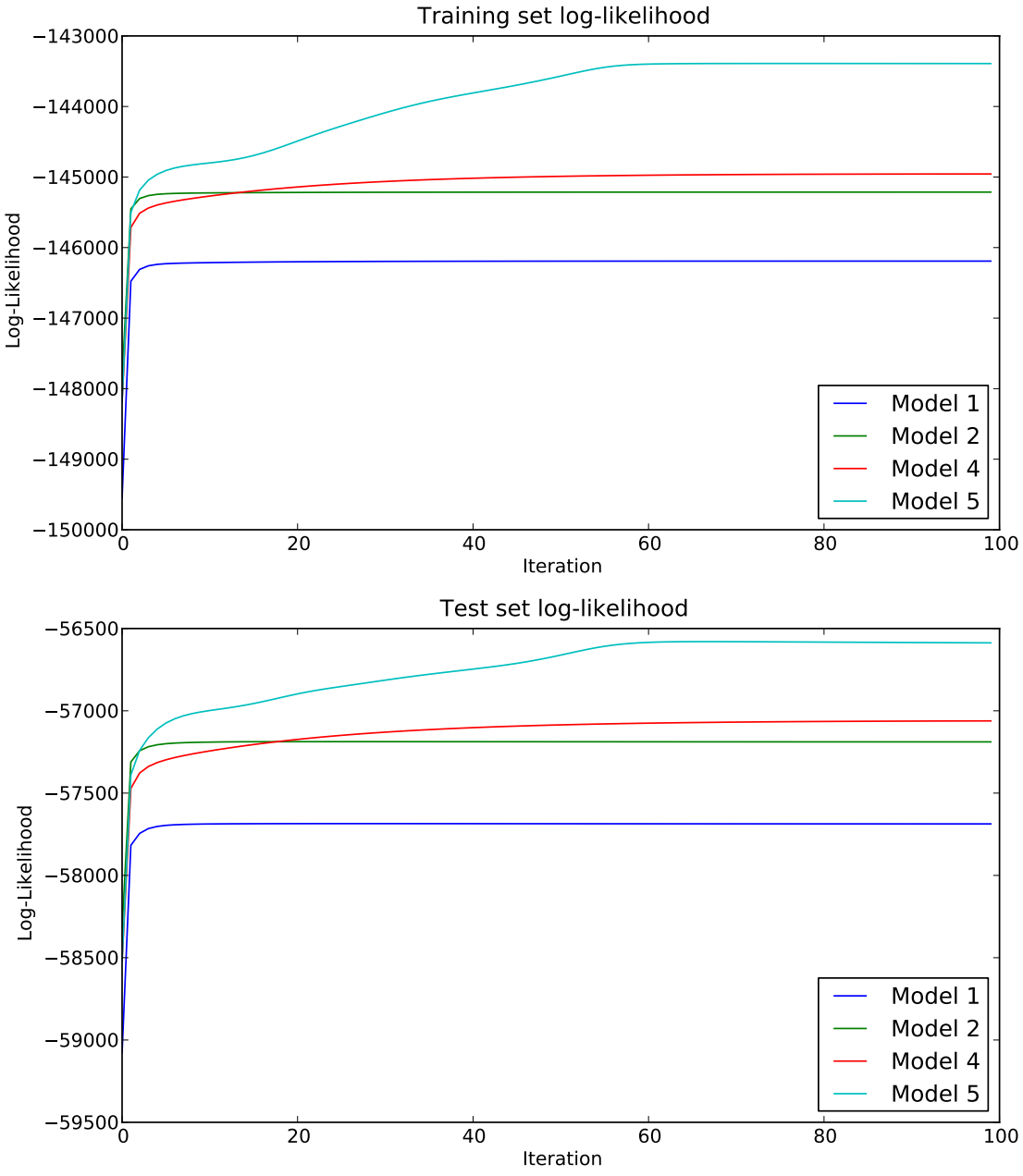


Figure 4.3: Log-likelihoods over EM iterations for the various models. Model 3 is far worse than anything plotted and is not shown.

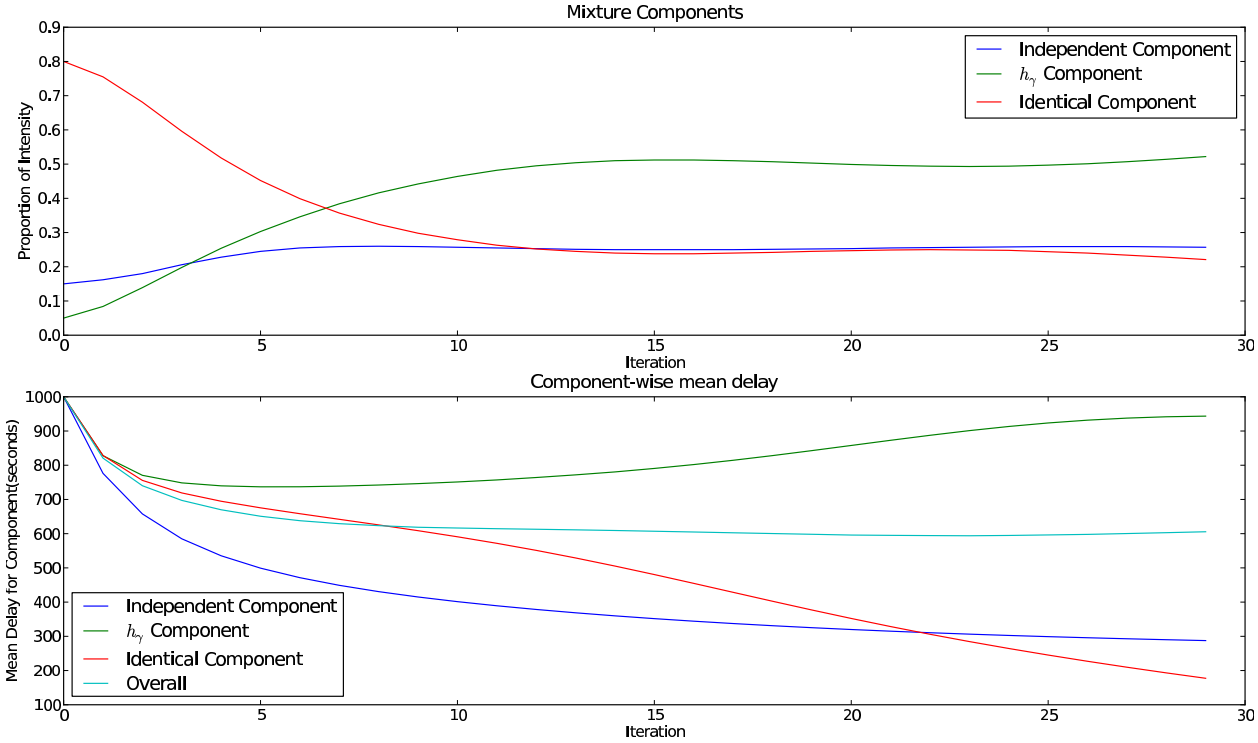


Figure 4.4: Trace of parameters of the individual mixture components in model 5.

delays. This can be seen in Figure 4.4 on page 48. The top plot shows the proportions of each component in the mixture, defined as the ratio of the average fertility to the total fertility. The dependence of fertility on the parameters is not shown, as it's very multi-dimensional. The bottom plot demonstrates that while the mean delay of the overall mixture remains almost constant throughout the EM iterations, different individual components have substantially different delay means.

4.6 Conclusion

Table 4.2 on page 49 reports the results for a cascade of models of increasing sophistication, demonstrating the gains that result from building up to the final model. The first stage of improvements, from the homogeneous to the periodic baseline and then to the independent transition model focuses on the times at which the events occur, and shows that roughly equivalent gains follow from modeling periodicity and from further capturing less periodic variability with essentially an exponential moving average. The big boost comes from a better labeling distribution that allows the features of events to depend on the previous events, capturing both the topic-wise hot trends and specific conversations.

Of course, the shape of the induced Poisson process has an effect. The different types of

Type	Train	Test
Homogenous Baseline Only	-167810.99	-66050.46
Periodic Baseline Only	-164695.23	-64758.70
Exp Delay, Identity transition	-146558.75	-57810.21
Exp Delay, Independent transition	-161905.96	-63017.34
Single intensity, Exp Delay, h_γ transition	-145752.84	-57383.06
Exp Delay, h_γ transition	-145557.26	-57313.59
Shared intensity, shared Exp delay, mixture transition	-145629.33	-57379.44
Mixture of (intensity, exp delay, different transitions)	-145152.76	-57130.91
Mixture of (intensity, gamma delay, different transitions)	-144621.58	-56966.75

Table 4.2: Log-likelihoods for models of increasing sophistication.

transitions have distinctly different estimated means for their delay distributions, which is to be expected since they capture different effects. In mixture, as seen in 4.2, the overall-intensity proxying independent transition has the highest mean, since the level of activity, averaged over labels, changes slower than the activity for a particular stock or topic. For shape, lower k, higher-variance Gamma distributions work best.

The final component is a fertility model that depends on the features of the event and allows some events to cause more successors than others. This actually has less impact on the log-likelihood than the rest of the components of the model, a trend that will continue in a different application in the next chapter.

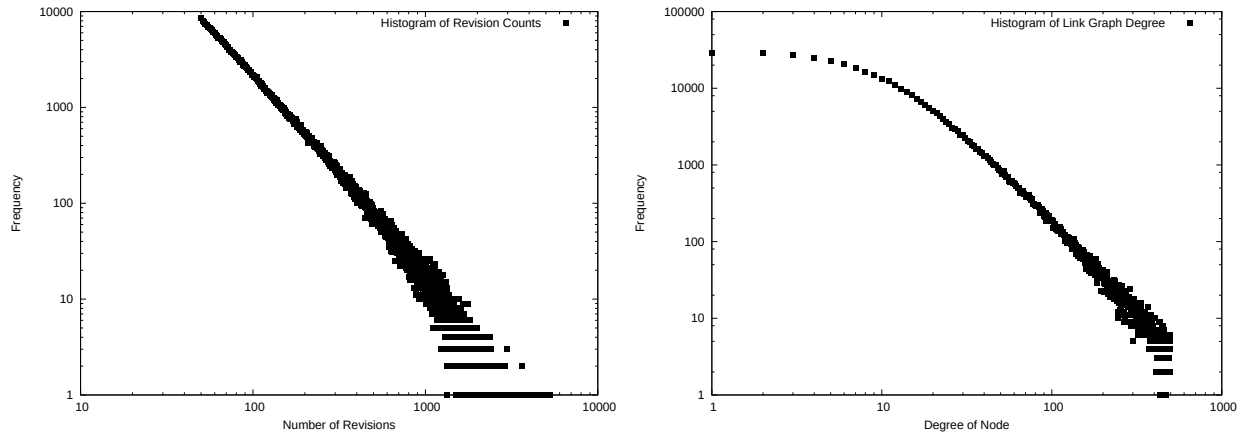
Chapter 5

Application: Wikipedia

Wikipedia is a public website that aims to build a complete encyclopedia. A key aspect differentiating it from traditional encyclopedia projects is that the content is provided by its users and may be edited by anyone at any time. Past revisions are recorded and the whole revision history is available for download. This historical record allows us to analyze the evolution of Wikipedia and explore how modifications of single pages cause cascades of other modifications to occur. In this chapter, we present the methodology and results of building a model of the revision history of Wikipedia. A key difference between this task and the work described in the earlier chapters, as well as the event modeling performed in prior work is the size of the dataset – we consider a vast amount of data and are able to successfully scale.

For clarity, lets first make concrete some terms used for the remainder of this chapter. Wikipedia is a collection of **pages**, each of which contains information about a particular topic. A page is composed of one or more **revisions**, which is a timestamped document representing the state of the page at a particular time. When a user modifies a page, a new revision, containing the modified body of the page is created. Additionally, each revision has associated meta-information such as the time at which the revision was made, the identity of the user making the revision and the comment left by the user describing the edit.

Each revision of a page can also contain some links to other pages within Wikipedia. At any moment in time, there exists a **link graph**, in which each page is a node and each link is a directed edge from the page containing the link to the page being linked to. Typically (as suggested and enforced by the project guidelines), each page has in-degree and out-degree of greater than zero. We will define a page's **neighborhood** as the collection of that page and all other pages either linking to or being linked from that page. In addition to the links that point to other Wikipedia pages, there exist other links to the wider web but for the purposes of this work, we do not consider them in the link graph.



Attribute	Value
Total Number of Pages	414,540
Total Number of Revisions	71,073,739
Average Number of Revisions	171.45
Number of Edges in Link Graph	7,559,283
Average Degree in Link Graph	18.23

Figure 5.1: Properties of the dataset (log-log plots). Note that we are restricting ourselves to normal pages with at least fifty pages, which explains the sharp cutoff on the revision histogram.

5.1 Dataset Description

As data, we will consider all normal Wikipedia pages (pages that are not category listings, user homepages, etc) that have at least fifty revisions in the dataset. Simply manipulating a dataset of this size is a non-trivial computational endeavor and performing learning requires distributed computation. All the computations described here were performed using Hadoop, an open source implementation of the MapReduce model for distributed computation[Dean and Ghemawat, 2004].

Figure 5.1 depicts histograms of the revision counts and link counts for the pages under consideration. By inspecting the graphs, it can be noted that for node degrees above 10 and, ignoring the discretization effects, both the link counts and revision counts appear to follow a power law distribution. This, in addition to being an interesting fact about Wikipedia, is relevant because the average link counts and revision counts don't tell the whole story about the size of some individual pages and their neighborhoods: there exist some pages for which there is both a large number of revisions and a large number of neighbors and these pose significant computation challenges.

We aim to build a probabilistic model for predicting edits of a page, based on the neighborhood of that page. Causes outside of the neighborhood are not considered for two reasons. Firstly, it's very unlikely for edits in two unrelated pages to related and if all pairs are

considered, there would be an immense number of false positive relationships, necessitating very strong regularization of non-adjacent pages. There is insufficient number of revisions in Wikipedia to accurately identify very subtle relationships; as the results in the end of this chapter show, even characterizing the relationships between different neighbors of a page is a difficult task. Furthermore, from a computational standpoint, considering all edits as potential causes for all other edits leads to such an immense number of possible relationships that even the parallel implementation on a large cluster is unable to cope.

5.2 Structure in Wikipedia's History

As we strive to build up a probabilistic model for the edits, it's useful to consider what kind of structure appears in the data that we would like the model to capture.

For the purposes of this work, edits can be broadly categorized into one of several types:

- **Minor Fixes**: these small tweaks include correcting the spelling of a single word, turning a normal word into a link, changing the display text for a link, adding or removing punctuation, etc. Only one or a few words in the document are affected.
- **Major Insert**: The insertion of a large amount of text. Often, the text happens to be migrated from a different page. This kind of edit is characterized by the addition of many words and the removal of none or very few. From the user's perspective, this corresponds to typing or pasting in a body of text with minimal editing of the context.
- **Major Delete:** The opposite of a major insert. Often performed by vandals who just delete a large section of the page.
- **Major Change**: An edit that affects a significant number of words but is not a simple insert or delete.
- **Revert:** Any edit that reverts the content of the page to a previous state. Most often, this is the immediately previous state but sometimes, it goes further back. These are often done as a response to vandalism, though edits done in good faith can be reverted as well.
- **Other Edit**: A change that affects more than a couple of words but is not a major insert or delete.

5.3 Delay Distributions

As in previous chapter, there exist many possible ways of parametrizing the duration between a revision and an event caused by that revision. In this application, each event will have

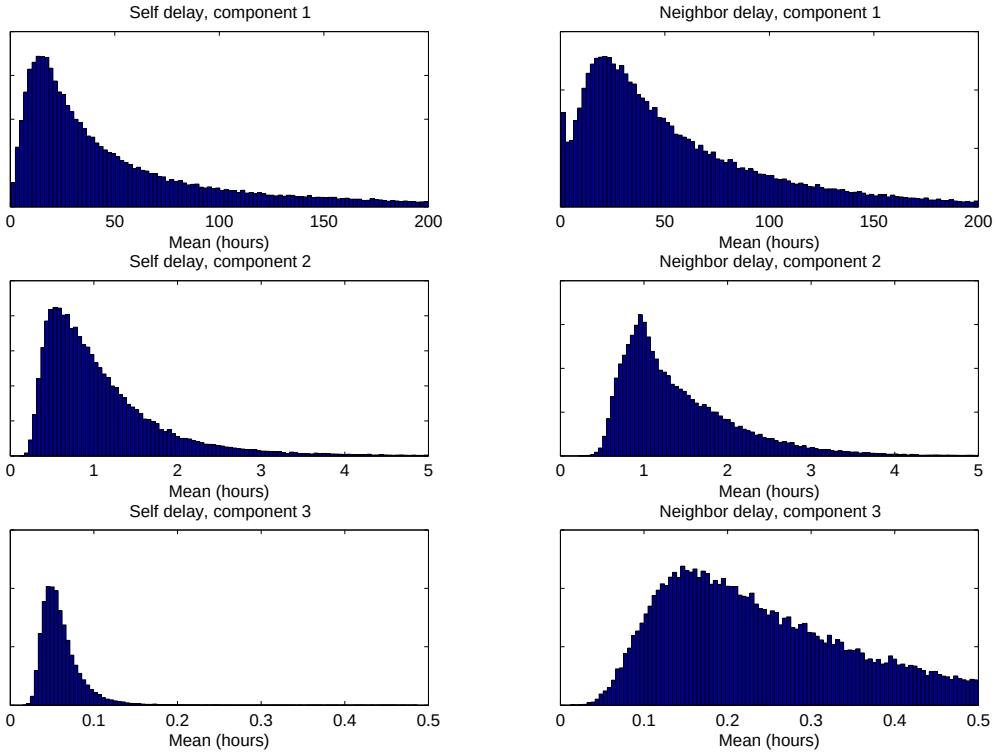


Figure 5.2: Delay distribution histogram over all pages. Each entry in a histogram is the mean of a component of delay distribution for one of the pages in the dataset. The left column contains histograms for the slow, medium and fast components of the same-page delay distribution, while the right column report the delay for neighboring pages.

a large number of possible causes, as pages have numerous neighbors so the intensity of the Poisson process at each event will be the sum over many possible triggers. This means that the exact shape of the distribution is not as important as in cases when only a few possible triggers are considered. We model the delay as a mixture of three exponential, with the intent of having each one capture a specific kind of editing phenomena. While this underlying intent motivated the design, as in mixture models in general, it would be a mistake to read too deeply into the components and assign each one a particular semantic meaning. For each page, we estimate both the parameters and the mixing weights of the components, with Figure 5.2 on page 53 showing a histogram of the estimated means.

The first is a high-mean exponential that corresponds to passer-bys who happen to discover the new content in the course of surfing. Among all pages, the average mean of this component is 48.5 hours for same-page delays and 53.2 hours for the between-page delay. A major function of this component is to act as a longer-term proxy for interest in a particular topic that may be popular due to some external recent event.

Another is an intermediate-mean exponential (averaging 1.08 hours for same-page and 1.41 for between-page). This is intended to correspond to users noticing a specific change and altering the page as a response, as well as a time-of-day phenomena.

The remaining component is a very fast response, with an average of 3.6 minutes for the same-page and 13.8 minutes for the adjacent-page delay. On the same-page, it captures edits that are actually caused by one another, either as an individual is making multiple modifications and saving the page along the way, or as a different user noticing the revisions on a news feed and instantly responding by changing or undoing them.

5.4 Transition Distribution

The intent of applying this model to Wikipedia is to capture the dynamics of edits and using the temporal revision data to identify structure in the data. This means the model needs to capture the significant attributes of the revision, in addition to its timestamp. However, we do not wish to develop a probability distribution over all the properties of the edit, as that would impractical. Attempting to model exactly which word is present in the edit, and exactly which contributor will make the edit is very complex and would dominate the likelihood. Instead, we will identify some key features of the edits and work to build a distribution over events as described by those features, not the raw edits. The features we use identify the type of edit (whether it's a major deletion, a small revision, an added link, etc), whether it was made by a known user and the identity of the page that the edit occurs on.

When a page with features x causes an event, the features of that event (which we will call x') must be drawn from some distribution over possible features. This conditional distribution needs to be learned, and in Chapter 4, we use a single-parameter parameterization of the conditional distribution h_γ that assumes independence between features.

However, when there are only a few possible combinations of features, in relation to the size of the training dataset, such independence assumptions are unnecessary and may be done away with. Furthermore, as the results will show, the resulting transition distribution will greatly depend on the interaction between features.

For this domain, however, the number of possible feature combinations is very large, as the identity of the page that the revision occurs on is part of the featurization: it's pointless to predict that a revision of a specific type will occur unless the identity of the page being revised is also predicted. We will partition the features into two categories $x = (x_1, x_2)$ where x_1 are features that can appear in all pages, whereas x_2 is just the identity of page. x_1 contains such information as the type of edit (from the categories defined in Section 5.2) and whether the contributor of that edit is logged in or anonymous.

Since the features of x_1 can appear in all pages, a massive amount of statistical information about them is available and it's possible to explicitly learn the transition matrix $P(x_1'|x_1)$ without imposing independence assumptions.

However, different pages may have a different transition structure; for example, the more controversial pages are more likely to have reversions. So, a single transition matrix is inadequate. We will model it as

$$\begin{aligned} x_1'|x_1, x_2 &\sim \text{Multinomial}\left(\theta_{x_1,x_2}\right) \\ \theta_{x_1,x_2} &\sim \text{Dirichlet}\left(\gamma_{x_1}\right). \end{aligned}$$

Through Dirichlet-Multinomial conjugacy, it's easy to compute $P(x_1'|x_1, x_2, x_{train})$ as it corresponds to MLE estimates for θ_{x_1}shrunken towards γ_{x_1}. For a particular page, when few transitions from (x_1, x_2) are observed, the probability is largely determined by the hyperparameters, which are tuned to reflect population means across all pages. As more transitions are observed, however, that page's transition probability becomes more driven by the specific observed probabilities on that page.

Features in x_2, describing the identity of the page are sufficiently rare and can take on sufficiently many values to make learning the full matrix impractical, so we must fall back to assuming a structured dependence between them and x_1. Our approach is to have each x_2 have its own transition matrix between x_1' and x_1 but having those transition matrices regularized towards a common shared one.

Figure 5.3 on page 56 shows log-likelihoods of successive iterations of the model. The versions with the regularized transition use the MAP estimate of θ_{x_1,x_2} and it can be seen that this regularization leads to uniformly superior likelihood. A baseline that does not allow events to cause other events is not shown, as its likelihood is much lower than the range of the plot. For this section, we will fix a specific page $x_2^\star$ that the events are occurring on and will write the Poisson intensities for just that page. The aggregate process over all pages will be a superposition of the individual processes over each page. The pairs of bars correspond to the following models:

- **No Neighbors**: The revisions on each page can be caused either by the baseline or a previous revision on that page but not by revisions of the neighbors:

  $$k_{(t,x)}^{x_2^\star}\left(t', x'\right) = \mathbf{1}\left(x_2 = x_2^\star\right) \alpha g(x'|x) h(x, x', t' - t)$$

  where α is a scalar, h a parametric mixtures estimated by ML and g_1 is an explicit transition matrix, estimated by ML but regularized as described above.

- **Neighbors, Same Transition**: Revisions of the page's neighbors in the link graph cause a Poisson process of edits on the page. That process has a its own delay distribution and intensity, but those are the same for all neighbors. The transition conditional distribution is the same both for events

  $$\begin{aligned} k_{(t,x)}^{x_2^\star}\left(t', x'\right) &= \mathbf{1}\left(x_2 = x_2^\star\right) \alpha_s g(x'|x) h_s(t' - t) \\ &+ \mathbf{1}\left(x_2 \in \delta x_2^\star\right) \alpha_n g(x'|x) h_n(t' - t). \end{aligned}$$

  Parameters for functions with with different subscripts are estimated separately.

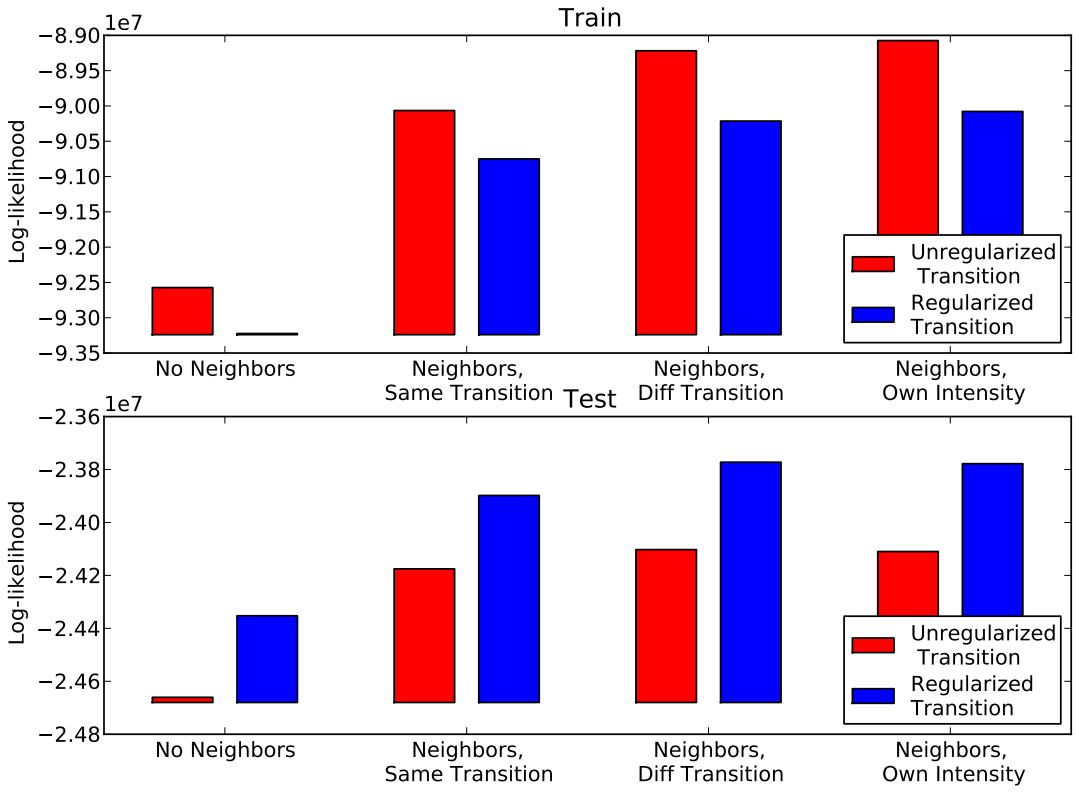


Figure 5.3: Log-Likelihoods of various models. Note that the models with regularized transition matrices perform significantly better on unseen data, but significantly worse on the training set; this means that the regularization is quite strong. The baseline-only is not shown but has $-1,48 \times 10^8$ training and -3.98×10^7 test log-likelihoods.

- **Neighbors, Different Transitions**: Revisions of the page's neighbors in the link graph cause a Poisson process of edits on the page. That process has a its own delay distribution and intensity, but those are the same for all neighbors. The transition distribution is the same both for events. The formulation is the same as above, only

$$\begin{aligned} k^{x_2^\star}_{(t,x)}(t', x') &= \mathbf{1}(x_2 = x_2^\star)\, \alpha_s g_s(x'|x) h_s(t' - t) \\ &+ \mathbf{1}(x_2 \in \delta x_2^\star)\, \alpha_n g_n(x'|x) h_n(t' - t). \end{aligned}$$

  Here, the parameters for the two different g are estimated separately and each is regularized towards γ same or γ neighbor respectively.

- **Neighbors, Own Intensities**: The same as above, expect that

$$\alpha(x, x') = \mathbf{1}(x_2^\star = x_2', x_2 \text{neighbor of } x_2^\star)\, \alpha_{x_2}.$$

where the each neighbor has its own intensity for causing events of $x_2^\star$. However, for all but the pages with the most revisions, there is insufficient data to estimate the individual αs accurately. Various regularizations are discussed later, but for the plot, the fixed-mixture is used.

5.4.1 Learned Transition Matrices

Figure 5.4 on page 58 shows the transition distribution matrix that is estimated. Instead of plotting the intensity, this figure shows the logarithm of the ratio between the corresponding entry in the transition matrix and the population probability of the observed feature. If this ratio is high, those labels of the caused events are much more likely than it would be otherwise.

The top row represents the intensity for the baseline, the labels of events whose cause is not a previous event. In the simple baseline-only model, without previous revisions causing successive ones, this would be all zeros since every observed event would have to be accounted for by the baseline. Here, positive values correspond to event types that the events-causing-events aspect of the model is less effective in capturing and thus are over-represented in the otherwise-unexplained column. Reverts, made both by known and anonymous contributors, are significantly underrepresented in this column, explaining that the rest of the model is effective in capturing them. Furthermore, for the rest of the possible event types, revisions made by known contributors are under-represented, as the rest of the model captures them better than the edits made by anonymous contributors. Events generated from this row account for 23.87% of total observed events.

The next block corresponds to edits on neighbors causing revisions of the page under consideration and are responsible for 19.11% of observed events . Here, the diagonal is predominantly positive, indicating that an event of a particular type on a neighbor makes an event of the same type more likely on the current page. There is a significantly positive rectangle for transitions between massive inserts, deletions and changes, though the magnitude of the ratio is almost identical in the rectangle. This means that significant modifications in text on the neighbors induce large modifications on the page under consideration, but the specific type of modification, or whether it's made by a known user are irrelevant. Large changes act as indications of interest in the topic or significant structural changes in the related pages.

The remaining block represents edits on a page causing further changes on the same page and is responsible for 57.02% of the observations. There is a stronger positive diagonal component here than above, as similar events tend to co-occur. Large changes lead to an over-representation of reverts following them, though large changes made by anonymous users are significantly more likely to lead to reverts. On the other hand, reverts produce extra large changes, as large modifications are made, reverted and come back again feeding an edit war. Note that reverts actually over-produce reverts. This is not primarily a first order effect (as the successive reverts don't often undo the previous undo), but rather captures controversial moments. The presence of a revert is an indication that previously, an unmeritorious edit

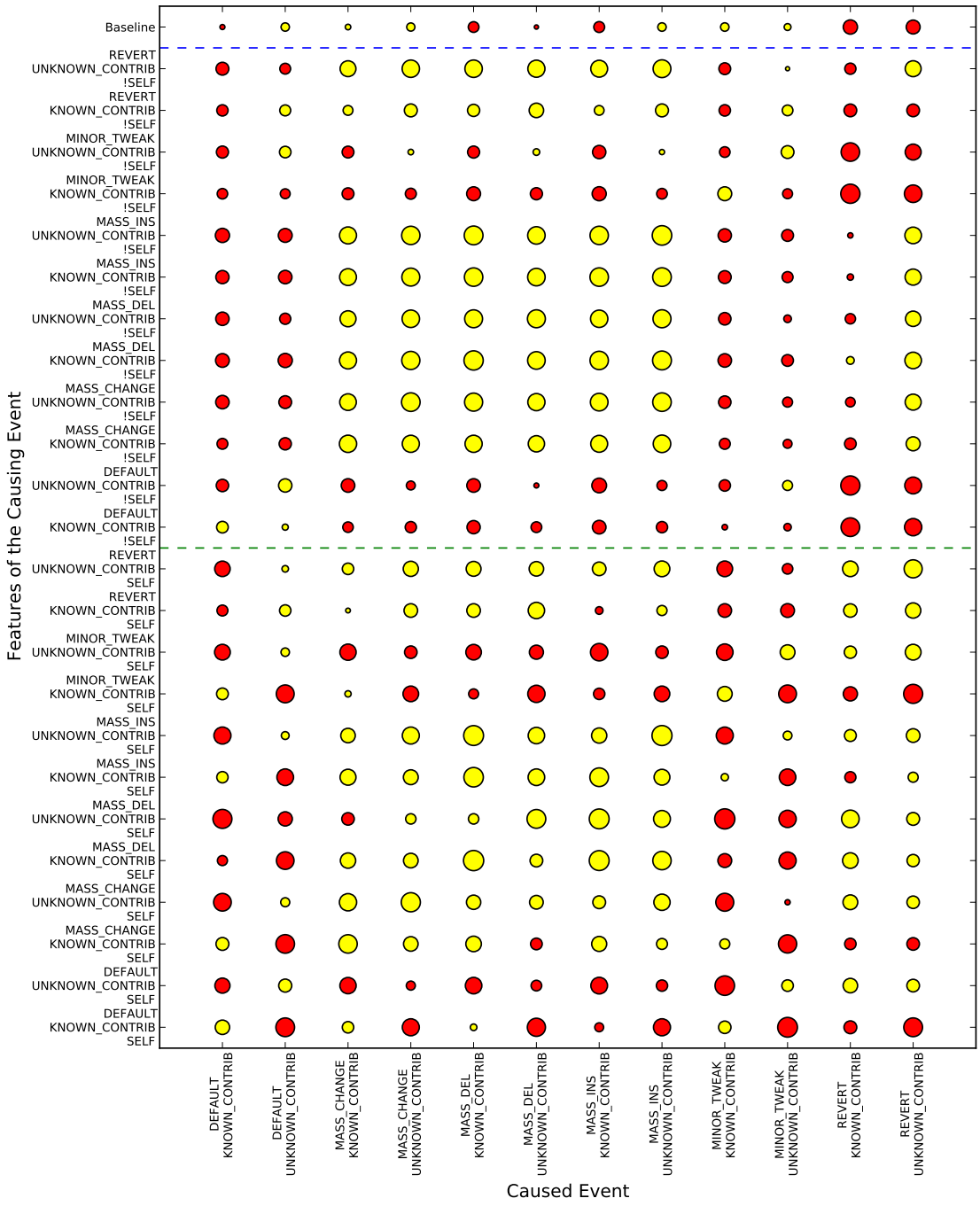


Figure 5.4: Learned Transition Matrix. The area of the circles corresponds to log of the conditional probability of the observed feature, divided by the marginal. The yellow, light-colored circles correspond to the transition being more likely than average; the red correspond to it being less likely.

was made, which suggests that future unmeritorious edits(that tend to be long and spammy) that need to be reverted are likely.

5.5 Regularizing Intensity Estimates

When, an edit occurs on a neighbor of a page $x_2^\star$, one would expect the identity of the neighbor to affect the likelihood of it causing an event on $x_2^\star$. This is because relationships between pages can differ, and while some neighbors in the link graph are only marginally related, others are very close. However, as it turns out, effectively estimating the intensities between a pair of pages is impractical unless a very large number of revisions have been observed. Even in the high-data regimes, strong regularization is required.

For a particular $x_2^\star$ and its neighbors, we need to estimate the individual α_{x_2}. For the moment, lets ignore the mixture aspects and assume that we know the true hard assignments of causes. Let k be the number of neighboring pages. The i^{th} page has occurred m_i times and has produced n_i events on the current page. Then, the terms of the log-likelihood that depend on α are

$$\sum_{i=1}^{k} -m_i\alpha_i + n_i \log \alpha_i$$

to which we add a regularize term $R(\alpha_1, \ldots, \alpha_k)$. Some possible regularizations are:

- L^1 regularization towards 0, $R = -\lambda \sum |\alpha_i|$ leading to $\hat{\alpha}_i = \frac{n_i}{m_i+\lambda}$. Note that since the objective here is log-likelihood and not L^2 loss, this leads to a shrunken but not sparse estimate.
- L^1 regularization towards a common γ,

  $$R = -\inf_\gamma \lambda \sum_i |\alpha_i - \gamma| = -\lambda \sum_i |\alpha_i - \text{median}(\alpha_1, \ldots, \alpha_k)|$$

  which is optimized by shrinking each $\hat{\alpha}_i = \frac{n_i}{m_i + \text{sign}(\hat{\alpha}_i - \text{median})\lambda_i}$, and noting that a possible minimizer is $\hat{\alpha}_i = \text{median}$.
- Both of the above can also be done as L^2 regularization, which leads to $\sum_i (m_i - n_i/\hat{\alpha}_i + 2\lambda\hat{\alpha}_i)$ or $\sum_i (m_i - n_i/\hat{\alpha}_i + 2\lambda(\hat{\alpha}_i - \bar{\alpha}_i))$, which are solvable for $\hat{\alpha}$ as quadratic equations.

Without knowing the correct assignments, it's possible to get regularized estimates through EM, where n_i are replaced by the sum of the soft assignments. However, these regularizers empirically lead to poorer likelihoods than simply using a single scalar α for all the neighbors, suggesting that there is not enough data to accurately estimate the individual αs. One

Page	Intensity	Page	Intensity
AH-64 Apache	**0.49**	**South Pole**	**0.46**
AH-1 Cobra	0.063	Equator	0.017
CH-47 Chinook	0.040	Roald Amundsen	0.016
101st Airborne Division (United States)	0.040	Ernest Shackleton	0.016
Mil Mi-24	0.037	Geography of Norway	0.015
Flight simulator	0.037	Navigation	0.015
List of Decepticons	0.034	South Georgia and the South Sandwich Islands	0.014
Tom Clancy's Ghost Recon Advanced Warfighter	0.034	National Geographic Society	0.014
Command & Conquer (video game)	0.033	List of cities by latitude	0.014

Table 5.1: Sample list of pages (in bold) and the intensities estimated for them and their neighbors. This is under strong regularization, which explains the similarity of the weights.

possible reason is that pages with a large number of events also have a large number of neighbors, so the estimation is always in a difficult regime. Furthermore, the hypothetical 'true' values of these parameters will change with time, as new neighbors appear and change.

One approach that works in high-data regimes is to let

$$\hat{\alpha}_{iREG} = \lambda \frac{\sum n_j}{\sum m_j} + (1 - \lambda) \frac{n_i}{m_i},$$

an average between the aggregate and individual maximizers. On a subset of the Wikipedia graph that includes only pages with more than 500 revisions, this improves withheld likelihoods over having a single α for all the neighbors. Figure 5.5 on page 61 shows that the improvement is very small, certainly smaller than the impact of other aspects of the model. Example pages and intensities estimated for their neighbors are shown in Table 5.1 on page 60. The regularizer λ is set to 0.8, forcing the lower weights to clump as each is lower-bounded by $0.8 \sum n_j / \sum m_j$. This has the effect of allowing a particularly-related neighbor to have a non-trivially higher intensity while preventing the less relevant neighbors from having a significantly lower one. Since activity on neighboring pages can always have some impact on the page under consideration, in the very least as a proxy for interest in the general topic, imposing a floor on the intensity helps reduce the danger of neglecting relevant information.

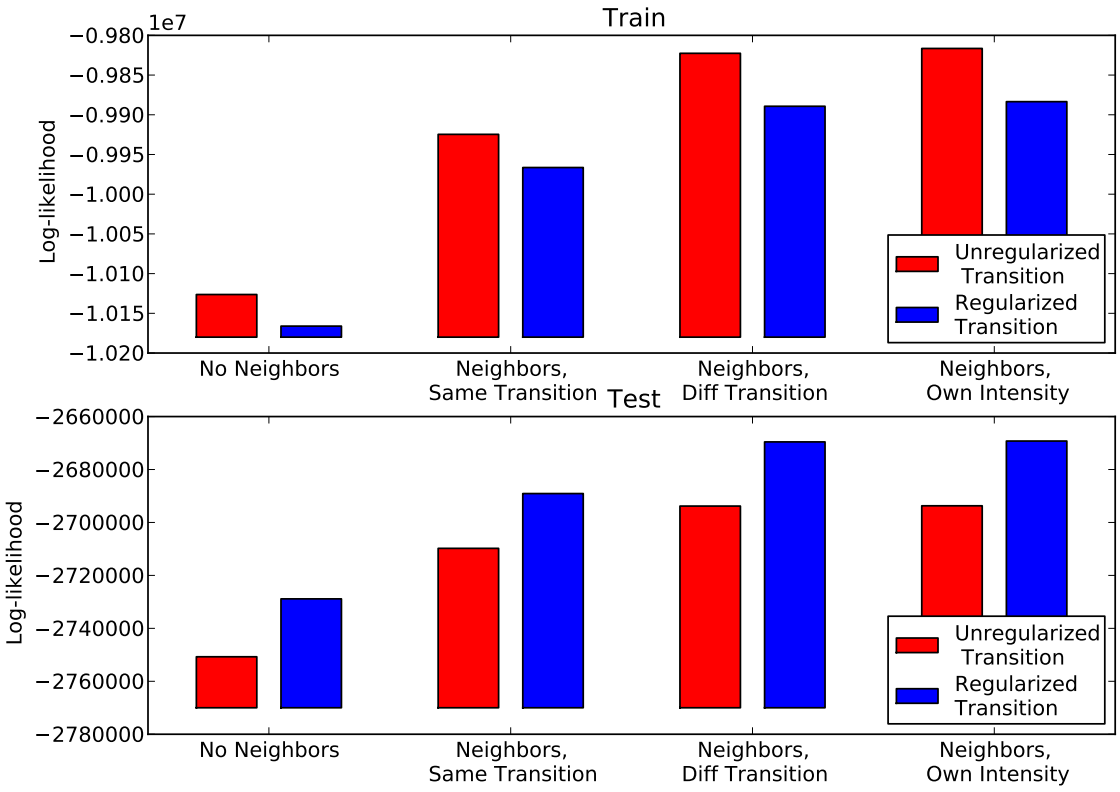


Figure 5.5: Log-Likelihoods on subset of data that includes only high revision count pages. The model with individual intensities has 104 higher test likelihood without regularizing the transitions, and 323 higher otherwise.

5.A Computational Details

As noted in Figure 5.1 on page 51, the number of pages and revisions in Wikipedia is sufficiently large to not fit on ram, or even storage, of a single machine. The raw dataset, exported from Wikipedia in 2008 is 2.8 terabytes. The reason that we use the 2008 version and not a more recent revision is that since then, no complete dump of Wikipedia's full revision history has been successful – the runtime required is multiple months and every attempt by Wikipedia to produce a distributable dump has failed mid-run. Decompressing the file takes more than 20 hours on a modern machine and recompressing would take more than 40 CPU days. While many of the pages in the full dump (such as user pages, or obscure pages with only one or several revisions) are discarded as the first step of pre-processing, the remaining amount of data is sufficiently large to continue posing problems.

Once the data is decompressed, which is unfortunately a serial operation due to the nature of the 7z compression format, all the following operations are performed using Hadoop

[Bialecki et al., 2005], an implementation of Dean and Ghemawat [2004]'s MapReduce running on the 398 node M45 cluster. Then, each page is separated, represented as differences between revisions and compressed.

A quick aside about MapReduce – this is a mode of distributed computation where the entities involved are key-value pairs. A map task is a function applied to each key-value pair that produces zero or more key-value pairs as output. Then, the resultant tuples go through the shuffle step, in which every value with the same key is aggregated. For each key, all the associated values are collected and presented to a reducer function, which again may produce zero or more key-value pairs, which can then be further processed or inspected as results. Since the map task for every key-value pair is independent, as is the reduce step for collections with different keys, the system can distribute the computation among multiple machines. One limitation of the current implementation of Hadoop is that in the map step, the value must fit in RAM. Furthermore, in the reduce step, all the values associated with every key must fit in RAM.

The computationally intensive step for EM is the collection of sufficient statistics, which is easily parallelized. In order to compute the zs for each event, it is necessary to consider all of the previous events that are potential causes. Lets consider a page p, composed of revisions $p_1, p_2, \ldots p_n$ and neighbor pages δp, containing revisions $r_1, r_2, \ldots, r_m$. One way to distribute the work is to to split the data into small pieces, each containing an event and all possible causes, so each datum would be composed of (p_i, {all of r_j recently before p_i}); we can then use the map function to compute the EM z variables for each event. However, this leads to massive duplication of data since every event is duplicated for every event it may have triggered and is impractical. A better approach is to make each datum contain the whole of $(p, \delta p)$ – a page (with all its revisions) and its neighbors, which provide sufficient to calculate the zs for every event of that page. From a memory standpoint, $(p, \delta p)$ can be quite large as pages can have many neighbors, each with many revisions, so careful choice of representation and compression is required so that every datum fits in memory. If a page and its neighborhood do not fit in RAM, clearly a hybrid approach, where each datum contains a subset of p_i and all the requisite r_j is appropriate.

These tuples are built by mapping each page p to key-value pairs $(p \to l_1), (p \to l_2), \ldots, (l_1 \to p), (l_2 \to p)$, which leads to all incoming and outgoing neighbors of a node being collected in the reduce step as $(p \to \{\text{outLinks}, \text{inLinks}\})$. However, the tuples $(p \to \{\text{outLinks}, \text{inLinks}\})$ may not fit in RAM, so instead, the operation is performed with the pages replaced by their IDs and then, further processing is performed to ensure that every work unit can fit in machine memory before the (compressed) page contents are reattached to their IDs. From that point on, each iteration of EM can be performed with a single MapReduce job, where the map task computes the sufficient statistics and the reduce task accumulates them and maximizes parameters.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we have developed a framework for constructing probabilistic models of events and demonstrated applicability on three different domains. Chapter 2 describes the components of the model and provides algorithms for parameter learning. In Chapter 3, we apply these ideas to modeling traffic in a computer network based on passive observation and use hypothesis testing to identify dependencies among machines in the network. In Chapter 4, we construct a model of Twitter messages about stocks and experimentally explore a range of design choices for the various distributions used in the construction of these models. Finally, in Chapter 5, we analyse the revision history of Wikipedia and demonstrate the scalability of our approach to large amounts of data.

The model presented in this work is conceptually relatively simple but flexible, due to the wide range of delay, transition and fertility functions that can be used within the framework. It captures a key aspect of event data – one event may trigger other ones – that has been insufficiently addressed by existing tools and so, has applications in various real-life situations. The resulting models can be used to extract dependencies and properties of a process' dynamics. Furthermore, the approaches' scaling properties allow the analysis of very large amounts of data, enabling the analysis of massive datasets.

6.2 Future Directions

At the same time, the presented model has limits in the interactions it can represent and relies on several key independence assumptions to yield tractable inference. For some applications, these limitations are significant and they point to directions for continued investigation:

- **Latent Triggers.** In this thesis, only observed events and the baseline Poisson process can trigger an event's occurrence. When the root cause is unobserved, the model com-

pensates by making the first observed, resultant events the trigger for the remaining resultant events. Adding explicit support for latent, unobserved events would greatly expand the power of these models, but may require more expensive inference and sufficient amounts of observed events for soundness of the more complex model. Furthermore, the times and features of the latent triggers may be a value of interest in their own right. The ideas described in Wingate et al. [2009] may provide an fruitful direction.

- **Interaction with Time Series**. In this work, all the parameters of the model (other than the baseline measure) can not change with time, leading to a stationary process. Incorporating support for transitions that either depend on the time at which the event occurs or a latent timeseries is another logical extension of the framework. For example, the number of expected triggered events can be a draw from a Gaussian process.

- **Bayesian Version**. The parameters to the model are estimated by regularized maximum likelihood, which clearly shows limitations in poor generalization when the number of estimated parameters is very large. A Bayesian formulation, combined with efficient inference, may statistically scale better and degrade more gracefully if the amount of observed data is insufficient.

Bibliography

R. Adams, I. Murray, and D. MacKay. Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities. In *Annual International Conference on Machine Learning(ICML)*. ACM New York, NY, USA, 2009.

R. P. Adams, I. Murray, and D. J. C. MacKay. The gaussian process density sampler. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS*, pages 9–16. MIT Press, 2008.

V. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *SIGCOMM'07*, Aug. 2007.

P. Barham, R. Black, M. Goldszmidt, R. Isaacs, J. MacCormick, R. Mortier, and A. Simma. Constellation: automated discovery of service and host dependencies in networked systems. Technical Report MSR-TR-2008-67, Microsoft Research, 2008.

A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley. Hadoop: a framework for running applications on large clusters built of commodity hardware, 2005.

R. Bodnarchuk and R. Bunt. A synthetic workload model for a distributed system file server. *SIGMETRICS Perform. Eval. Rev.*, 19(1):50–59, 1991. ISSN 0163-5999. doi: http://doi.acm.org/10.1145/107972.107978.

D. Cox and V. Isham. *Point processes*. Chapman & Hall/CRC, 1980.

D. Daley and D. Vere-Jones. *An introduction to the theory of point processes: Volume I: elementary theory and methods*. Springer, second edition, 2003.

D. Daley and D. Vere-Jones. *An introduction to the theory of point processes: Volume II. probability and its applications*. Springer-Verlag, 2008.

J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *Symposium on Operating Systems Design & Implementation (OSDI)*, 2004.

A. Dempster, N. Laird, D. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1): 1–38, 1977.

W. Fischer and K. Meier-Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18:149–171, September 1993.

D. Freed and L. Shepp. A poisson process whose rate is a hidden markov process. *Advances in Applied Probability*, 14(1):21–36, 1982.

A. Ganapathi. Statistics-driven workload modeling for the cloud. Technical Report UCB//EECS-2009-160, University of California, Berkeley., 2009.

A. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83, 1971.

J. Kingman. *Poisson processes*. Oxford University Press, USA, 1993.

P. Lewis, G. Shedler, and N. P. S. M. CA. Simulation of nonhomogeneous poisson processes by thinning. 1978.

B. Lindsay. *Mixture models: theory, geometry, and applications*. Institute of Mathematical Statistics, Hayward, 1995. ISBN 0940600323.

U. Nodelman, C. R. Shelton, and D. Koller. Continuous time Bayesian networks. In *Uncertainty in Artificial Intelligence(UAI)*, pages 378–387, 2002.

U. Nodelman, C. R. Shelton, and D. Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. In *Uncertainty in Artificial Intelligence(UAI)*, 2005.

Y. Ogata. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83(401):9–27, 1988.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

S. Rajaram, T. Graepel, and R. Herbrich. Poisson-networks: A model for structured point processes. In *International Workshop on Artificial Intelligence and Statistics (AISTAT)*, 2005.

A. Rényi. Remarks on the Poisson process. In *Symposium on Probability Methods in Analysis*, pages 280–286. Springer, 1967.

M. Rudemo. Doubly stochastic poisson processes and process control. *Advances in Applied Probability*, pages 318–338, 1972.

T. Rydén. An EM algorithm for estimation in Markov-modulated Poisson processes. *Computational Statistics and Data Analysis*, 21:431–447, 1996.

S. Saria, U. Nodelman, and D. Koller. Reasoning at the right time granularity. In *Uncertainty in Artificial Intelligence(UAI)*. Citeseer, 2005.

S. Scott and P. Smyth. The markov modulated poisson process and markov poisson cascade with applications to web traffic data. *Bayesian Statistics*, 7:671–680, 2003.

A. Simma, M. Goldszmidt, J. MacCormick, P. Barham, R. Black, R. Isaacs, and R. Mortier. CT-NOR: Representing and reasoning about events in continuous time. In *Uncertainty in Artificial Intelligence(UAI)*, 2008.

J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):479–498, 2002.

A. Veen and F. Schoenberg. Estimation of space-time branching process models in seismology using an em-type algorithm. *Journal of the American Statistical Association*, 103(482): 614–624, 2008.

D. Vere-Jones. Some models and procedures for space-time point processes. *Environmental and Ecological Statistics*, 16(2):173–195, 2009.

D. Vere-Jones and R. Davies. A statistical survey of earthquakes in the main seismic region of new zealand. *NZJ Geol. Geophys*, 9:251–284, 1966.

J. Von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12:36–38, 1951.

L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer Verlag, 2004.

W. Wei, B. Wang, and D. Towsley. Continuous-time hidden Markov models for network performance evaluation. *Performance Evaluation*, 49(1-4):129–146, 2002.

D. Wingate, N. Goodman, D. Roy, and J. Tenenbaum. The infinite latent events model. In *Uncertainty in Artificial Intelligence(UAI)*, 2009.