Supplemental Material

Online Inference for the Infinite Topic-Cluster Model: Storylines from Streaming Text

Amr Ahmed , Qirong Ho , Choon Hui Teo , Jacob Eisenstein , Alexander J. Smola , Eric P. Xing

A Inference

We use C to denote a generic count of co-occurrences, for example, C_{tdk} is the number of words in document d at epoch t that are generated from topic k. We might remove a dimension to denote summation, for example, $C_{td.}$ is the total number of words in document d at epoch t and $C_{t.k}$ is the total number of words generated from topic k at epoch t. Finally, we use a negative sign in the superscript to denote exclusion, for example, C^{-tdi}_{tdk} is the same quantity as C_{tdk} without the contribution of word i, although sometimes we abuse notation and use i if the meaning is clear from the context.

Sampling z_{tdi}:

$$\begin{aligned} P(z_{tdi}=k|w_{tdi}=w, s_{td}=s, \text{rest}) &= \mathbb{E}_{p(\pi_s|\text{rest})}\Big[\mathbb{E}_{p(\theta_{td}|\text{rest})}[\theta_{tdk}|\pi_s]\Big]\mathbb{E}_{p(\phi_k|\text{rest})}\Big[\phi_{kw}\Big] \\ &= \mathbb{E}_{p(\pi_s|\text{rest})}\Big[\frac{C^{-i}_{tdk}+\alpha\pi_{sk}}{C^{-i}_{td.}+\alpha}\Big]\frac{C^{-i}_{kw}+\phi_0}{C^{-i}_{k.}+\phi_0 W} \\ &= \frac{C_{tdk}+\alpha'_s\mathbb{E}_{p(\pi_s|\text{rest})}[\pi_{sk}]}{C^{-i}_{td.}+\alpha}\frac{C^{-i}_{kw}+\phi_0}{C^{-i}_{k.}+\phi_0 W} \\ &= \frac{C^{-i}_{tdk}+\alpha\frac{C^{-i}_{sk}+\pi_{0,k}}{C^{-i}_{s.}+\sum_{k=1}^{K+1}\pi_{0,k}}}{C^{-i}_{td.}+\alpha}\frac{C^{-i}_{kw}+\phi_0}{C^{-i}_{k.}+\phi_0 W} \end{aligned} \tag{1}$$

Where, `rest` denotes all other variables not mentioned explicitly . The iterated expectation arises because the structure of the hierarchal prior, and each expectation is just the mean of the corresponding Dirichlet posterior.

Sampling s_{td}:

$$\begin{aligned} &P(s_{td}|\mathbf{s}^{-td}_{t-\Delta:t}, \mathbf{z}_{td}, \mathbf{e}_{td}, \mathbf{w}^{K+1}_{td}, \text{rest}) = \\ &\underbrace{P(s_{td}|\mathbf{s}^{-td}_{t-\Delta:t})}_{\text{Prior}}\underbrace{P(\mathbf{z}_{td}|s_{td},\text{rest})P(\mathbf{e}_{td}|s_{td},\text{rest})P(\mathbf{w}^{K+1}_{td}|s_{td},\text{rest})}_{\text{Emission}} \end{aligned} \tag{2}$$

We begin with the prior:

$$P(s_{td}|\mathbf{s}^{-td}_{t-\Delta:t}) \propto \begin{cases} m'_{st}+m^{-td}_{st} & s \text{ is an already existing story} \\ \gamma & s \text{ is a new story} \end{cases} \tag{3}$$

For the emission term $P(\mathbf{e}_{td}|s_{td},\text{rest})$:

Let $C_{td,e}$ be the number of times entity e appears in document d, then the above probability is just the ratio between two partition functions.

$$
\begin{aligned}
P(\mathbf{e}_{td}|s_{td}=s,\text{rest}) &= \int P(\mathbf{e}_{td}|\Omega_s)p(\Omega_s|\Omega_0,\text{rest})d\Omega_s \\
&= \frac{\prod_{e=1}^{E}\Gamma\Big(C_{td,e}+C_{se}^{-td}+\Omega_0\Big)}{\Gamma\Big(\sum_{e=1}^{E}[C_{td,e}+C_{se}^{-td}+\Omega_0]\Big)}\frac{\Gamma\Big(\sum_{e=1}^{E}[C_{se}^{-td}+\Omega_0]\Big)}{\prod_{e=1}^{E}\Gamma\Big(C_{se}^{-td}+\Omega_0\Big)}
\end{aligned}
\tag{4}
$$

The above equation is valid for both the cases that s is an exisiting story or s is a new story. However, when s is a new story, C_{se}^{-td} is zero for all entities e.

For the emission term $P(\mathbf{w}_{td}^{K+1}|s_{td},\text{rest})$:

$\mathbf{w}_{td}^{K+1}$ is the set of words in document td generated from topic $K+1$, where topic $K+1$ denotes the story-specific topic $\phi_{s_{td}}$. Let $C_{td,w}^{K+1}$ be the number of times word w was generated from topic $K+1$ in document d, then the above probability is again just the ratio between two partition functions.

$$
\begin{aligned}
P(\mathbf{w}_{td}^{K+1}|s_{td}=s,\text{rest}) &= \int P(\mathbf{w}_{td}^{K+1}|\phi_s)p(\phi_s|\Phi_0,\text{rest})d\phi_s \\
&= \frac{\prod_{w=1}^{W}\Gamma\Big(C_{td,w}^{K+1}+C_{sw}^{-td}+\phi_0\Big)}{\Gamma\Big(\sum_{w=1}^{W}[C_{td,w}^{K+1}+C_{sw}^{-td}+\phi_0]\Big)}\frac{\Gamma\Big(\sum_{w=1}^{W}[C_{sw}^{-td}+\phi_0]\Big)}{\prod_{w=1}^{W}\Gamma\Big(C_{sw}^{-td}+\phi_0\Big)}
\end{aligned}
\tag{5}
$$

Again, the above equation works for both the cases when s is an exisiting or if s is a new story. However, when s is a new story, C_{sw}^{-td} is zero for all words w.

For the emission term $P(\mathbf{z}_{td}|s_{td}=s,\text{rest})$:

Unfortunately, we can not express this probability as the ratio of two partition functions. However, we note that $\mathbf{z}_{td}$ is exchangeable, hence we can evaluate this expression using a left-right calculation similar to Wallach (2008):

$$
\begin{aligned}
P(\mathbf{z}_{td}|s_{td}=s,\text{rest}) &= \prod_{i=1}^{n_{td}} P(z_{tdi}|s_{td}=s,\mathbf{z}_{td}^{-td,(n\geq i)},\text{rest}) \\
&= \prod_{i=1}^{n_{td}} \frac{C_{tdk}^{-td,(n\geq i)}+\alpha\frac{C_{sk}^{-td,(n\geq i)}+\pi_{0,k}}{C_{s\cdot}^{-td,(n\geq i)}+\sum_{k=1}^{K+1}\pi_{0,k}}}{C_{td\cdot}^{-td,(n\geq i)}+\alpha}
\end{aligned}
\tag{6}
$$

,where the superscript $-td,(n\geq i)$ means excluding all words in document td that came after position i. The second line in the above equation follows from (1). Moreover,the above equation is valid for a new story except that $C_{sk}^{-td}=0$.

For the future term $P(\mathbf{s}_{t+1:t+\delta}|\mathbf{s}_{t-\Delta:t+\delta}^{-td},s_{td}=s)$:

This factor is only multiplied to (Eq2) during the resampling phase. This factor represents the transition probability which measures the likelihood of the table assignments at future epochs if we choose to assign $s_{td}=s$. Now we focus on computing one of these probabilities at epoch $t+\delta$. With reference to the RCRP construction and considering that documents are exchangeable within each epoch, similar to Antoniak (1974), we have:

$$
\mathrm{P}(s_{t+\delta}|s_{t+\delta-\Delta:t+\delta-1}^{-td\to s}) = \gamma^{S_{t+\delta}^{born}}\frac{\prod_{v\in S_{t+\delta}^{born}}[1]^{m_{v,t+\delta}}\prod_{v\notin S_{t+\delta}^{born}}[m'^{,-td\to s}_{v,t+\delta}]^{m_{v,t+\delta}}}{\prod_{i=1}^{m_{\cdot,t+\delta}}(m'^{,-td\to v}_{\cdot,t+\delta}+\gamma+i)}
\tag{7}
$$

where $S_{t+\delta}^{born}$ is the number of stories born at epoch $t+\delta$,$m_{\cdot,t+\delta}$ is the summation of $m_{k,t+\delta}$ over the first dimension (stories), and $m'_{\cdot,t+\delta}$ is defined similarly. Finally, $[a]^c=a(a+1)\cdots(a+c-1)$.

A.1 Particle Importance Weight

The unnormalized importance weight for particle f after receiving document d at time t, $\omega^f_{1:t,d}$, where $(1:t,d)$ is a shorthand for all documents up to document d at time t, can be calculated recursively as follows. Let $\mathbf{x}_{td}$ denote the words and named entities of document d at time t, then:

$$\begin{aligned}\omega^f_{1:t,d} &= \omega^f_{1:t,d-1}\frac{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:td},\mathbf{s}^f_{1:td},\mathbf{x}_{1:t,d-1})P(\mathbf{z}^f_{td},\mathbf{s}^f_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})}{Q(\mathbf{z}^f_{1:td},\mathbf{s}^f_{1:td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:td})} \\ &= \omega^f_{1:t,d-1}\frac{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:td},\mathbf{s}^f_{1:td},\mathbf{x}_{1:t,d-1})P(\mathbf{z}^f_{td},\mathbf{s}^f_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})}{P(\mathbf{z}^f_{1:td},\mathbf{s}^f_{1:td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:td})} \end{aligned} \quad (8)$$

Using Bayes rule $P(A|B,D) = \frac{P(D|B,A)P(A|B)}{P(D|B)}$, we can re-write the denominator as:

$$\begin{aligned}P(\mathbf{z}^f_{td},\mathbf{s}^f_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1},\mathbf{x}_{td}) &= \frac{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{z}^f_{td},\mathbf{s}^f_{1:t,d-1},\mathbf{s}^f_{td},\mathbf{x}_{1:t,d-1})P(\mathbf{z}^f_{td},\mathbf{s}^f_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})}{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})} \\ &= \frac{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:td},\mathbf{s}^f_{1:td},\mathbf{x}_{1:t,d-1})P(\mathbf{z}^f_{td},\mathbf{s}^f_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})}{P(\mathbf{x}_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1})} \end{aligned} \quad (9)$$

Combining Eq. (8) and Eq. (9), we get:

$$\omega^f_{1:td} = \omega^f_{1:t,d-1}P(\mathbf{x}_{td}|\mathbf{z}^f_{1:t,d-1},\mathbf{s}^f_{1:t,d-1},\mathbf{x}_{1:t,d-1}) \quad (10)$$

The marginal probability in Eq.(10) can be computed using the last 10 samples from the document-localized MCMC loop in Algorithm 2.

A.2 Implementation and Storage

[1]

Implementing our parallel SMC algorithm for large datasets poses runtime and memory challenges. While our algorithm mostly works on single particles by spawning one computational thread per particle filter, it also has to copy entire particles during particle resampling (done via a master thread). Hence we require a *thread-safe* data structure that supports fast updates of individual particles' data, *and* fast copying of particles.

It should be obvious that the naive implementation, where each particle has its own set of arrays for storage, is very inefficient when it comes to particle resampling — in the worst case, we would have to duplicate all particle arrays element-by-element. Worse, our memory requirements would grow linearly in the number of particles, making large data streams impractical even for modest numbers of particles.

Inheritance trees Instead, we employ an idea from Canini *et al.* Canini et al. (2009), in which particles maintain a memory-efficient representation called an "inheritance tree". In this representation, each particle is associated with a tree vertex, which stores the actual data. The key idea is that child vertices inherit their ancestors' data, so they need only store changes relative to their ancestors, in the form of a dictionary or hash map. To save memory, data elements with value 0 are not explictly represented unless necessary (e.g. when a parent has nonzero value). New vertices are created only when writing data, and only under two circumstances: first, when the particle to be changed shares a vertex with other particles, and second, when the particle to be changed is associated with an interior vertex. In both cases, a new leaf vertex is created for the particle in question.

This representation dramatically reduces memory usage for large numbers of particles, and also makes particle replication a constant runtime operation. The tradeoff however, is that data retrieval becomes linear time in the depth of the tree, although writing data remains (amortized) constant time. This disadvantage can be mitigated via tree maintenance operations, in which we *prune* branches without particles and then *collapse* unnecessary long branches — refer to Figure 1 for an example. With tree maintenance, data retrieval becomes a practically constant time operation.

[1] This Section extends upon the description which appears on Ahmed et al. (2011) – it is added here for self-containment

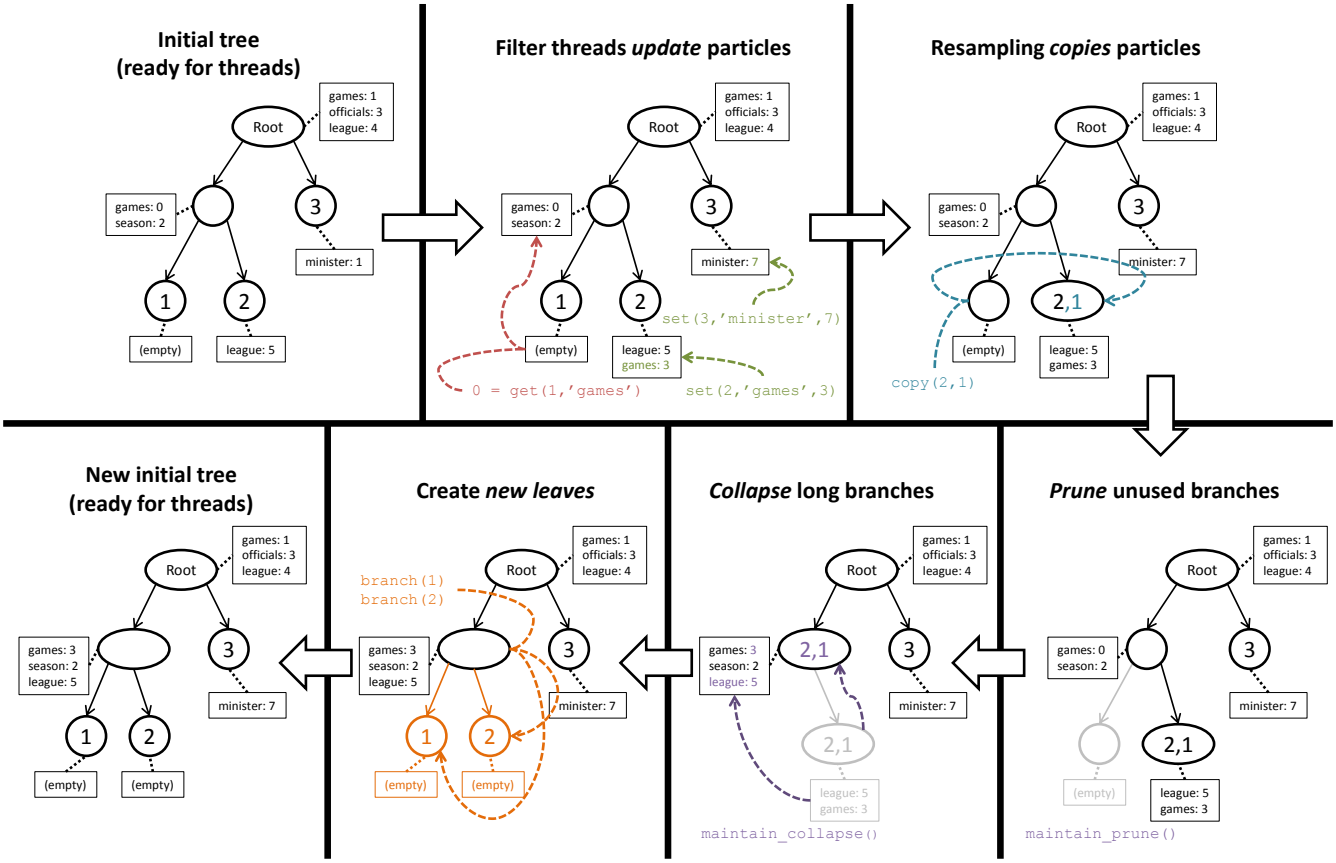

Figure 1: Inheritance tree operations in the context of our SMC algorithm. Numbers within a vertex represent associated particles. Each vertex's hash map is represented by a table, connected by a dotted line.

Thread safety Thus far, we have only described Canini *et al.*'s version of the inheritance tree, which is *not* thread-safe. To see why, consider what happens when particle 1 is associated with the parent vertex of particle 2. If a thread writes to particle 1 while another is reading from particle 2, it may happen that the second thread needs to read from the parent vertex. This creates a race condition, which is unacceptable for our parallel algorithm.

To ensure thread safety, we augment the inheritance tree by requiring every particle to have its own *leaf* in the tree. This makes particle writes thread-safe, because no particle is ever an ancestor of another, and writes only go to the particle itself, never to ancestors. Furthermore, every particle is associated with only one computational thread, so there will never be simultaneous writes to the *same* particle. On the other hand, data reads, even to ancestors, are inherently thread-safe and present no issue. To maintain this requirement, observe that particle-vertex associations can only change during particle resampling, which is handled by a a master thread. Immediately after resampling, we *branch* off a new leaf for every particle at an interior node. Once this is done, the individual filter threads may be run safely in parallel.

In summary, the inheritance tree has four operations, detailed in Figure 1:

1. `branch(f)`: creates a new leaf for particle f.
2. `get(f,i)`, `set(f,i,value)`: retrieve/write data elements i for particle f.
3. `copy(f,g)`: replicate particle f to g.
4. `maintain()`: prune particle-less branches and collapse unnecessary long branches.

Extended inheritance trees Our thread-safe inheritance tree supports most of our data storage needs. However, parts of our algorithm require storage of *sets of objects*, rather than integer values. For example, our story sampling equation needs the set of stories associated with each named entity, as well as the number of times each story-to-

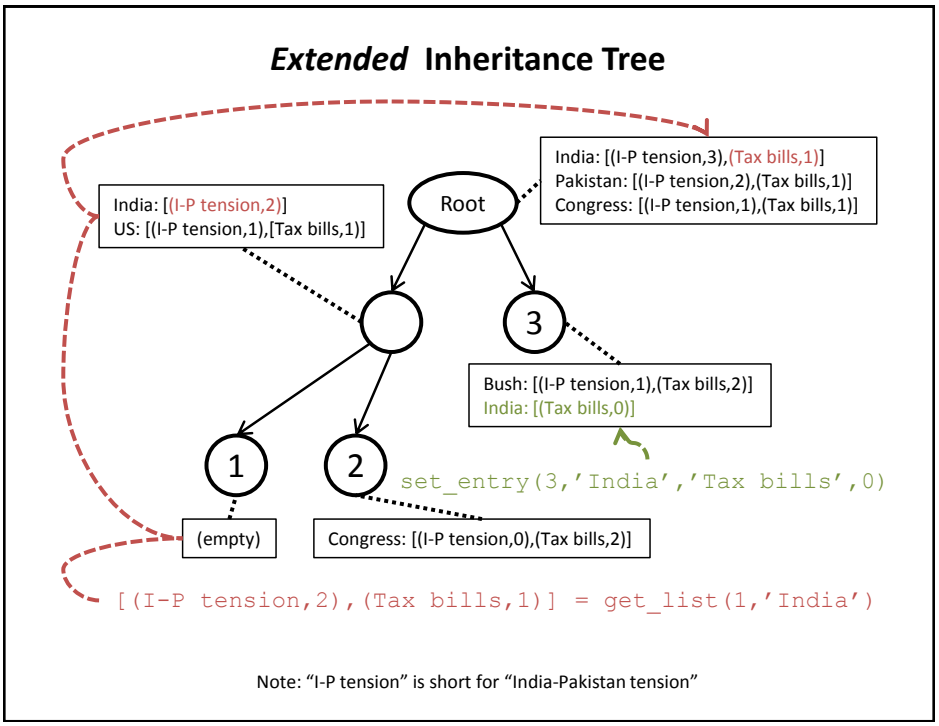

Figure 2: Operations on an *extended inheritance tree*, which stores sets of objects in particles, shown as lists in tables connected to particle-numbered tree nodes. Our algorithm requires particles to store some data as sets of objects instead of arrays, and this data structure allows (a) the particles to be replicated in constant-time, and (b) the object sets to be retrieved in amortized linear time.

entity association occurs.

We extend the basic inheritance tree by making its hash maps store *other hash maps* as values. These second-level hash maps then store objects as key-value pairs. Using the story sampling equation as an example, the first-level hash map uses named entities as keys, and the second-level hash map uses stories as keys and association counts as values (Figure 2 has an example). Observe that the count for a particular story-entity association can be retrieved or updated in amortized constant time.

Like the first-level hash maps, second-level hash maps only store changes relative to ancestor vertices, which keeps memory requirements to a minimum while facilitating constant-time particle replication. Obtaining the object set for a first-level entry i (a named entity in the previous example) thus requires going through second-level hash maps for entry i, over the current vertex and all its ancestors. This has runtime linear in the size of all relevant second-level hash maps.

Purging unnecessary data Because the RCRP prior only looks at a finite time window from $t - \Delta$ to t, we only need particle data from that window in memory, while antecedent data can be safely purged to disk. The same observation applies to the incoming word and entity streams; we only need data from that window in memory. These features keep our algorithm's memory usage constant if Δ is assumed constant, which is critical for online execution. Moreover, our bounded memory requirements allow us to handle datasets of the scale seen in our experiments.

References

Ahmed, A., Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo (2011). Unified analysis of streaming news. In *WWW*.

Antoniak, C. E. (1974). Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics 2*(6), 1152–1174.

Canini, K. R., L. Shi, and T. L. Griffiths (2009). Online inference of topics with latent dirichlet allocation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Wallach, H. (2008). Structured topic models for language. Technical report, PhD. Cambridge.