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

Amr Ahmed CMU
Qirong Ho CMU
Choon Hui Teo Yahoo! Labs
Jacob Eisenstein CMU
Alex J. Smola Yahoo! Research
Eric P. Xing CMU

Abstract

We present the time-dependent topic-cluster model, a hierarchical approach for combining Latent Dirichlet Allocation and clustering via the Recurrent Chinese Restaurant Process. It inherits the advantages of both of its constituents, namely interpretability and concise representation. We show how it can be applied to streaming collections of objects such as real world feeds in a news portal. We provide details of a parallel Sequential Monte Carlo algorithm to perform inference in the resulting graphical model which scales to hundred of thousands of documents.

1 INTRODUCTION

Internet news portals provide an increasingly important service for information dissemination. For good performance they need to provide essential capabilities to the reader:

Clustering: Given the high frequency of news articles — in considerable excess of one article per second even for quality English news sites — it is vital to group similar articles together such that readers can sift through relevant information quickly.

Timelines: Aggregation of articles should not only occur in terms of current articles but it should also account for previous news. This matters considerably for stories that are just about to drop off the radar so that they may be categorized efficiently into the bigger context of related news.

Content analysis: We would like to group content at three levels of organization: high-level topics, individual stories, and entities. For any given story, we would like to be able to identify the most relevant topics, and also the individual entities that distinguish this event from others which are in the same overall topic. For example, while the topic of the story might be the death of a pop star, the identity *Michael Jackson* will help distinguish this story from similar stories.

Online processing: As we continually receive news documents, our understanding of the topics occurring in the event stream should improve. This is not *necessarily* the case for simple clustering models — increasing the amount of data, along time, will simply increase the number of clusters.Yet topic models are unsuitable for direct analysis since they do not reason well at an individual event level.

The above desiderata are often served by *separate* algorithms which cluster, annotate, and classify news. Such an endeavour can be costly in terms of required editorial data and engineering support. Instead, we propose a *unified* statistical model to satisfy all demands simultaneously. We show how this model can be applied to data from a major Internet News portal.

From the view of statistics, topic models, such as Latent Dirichlet Allocation (LDA), and clustering serve two rather incomparable goals, both of which are suitable to address the above problems partially. Yet, each of these tools in isolation is quite unsuitable to address the challenge.

Clustering is one of the first tools one would suggest to address the problem of news aggregation. However, it is deficient in three regards: the number of clusters is a linear function of the number of days (assuming that the expected number of stories per day is constant), yet models such as Dirichlet Process Mixtures (Antoniak 1974) only allow for a logarithmic or sublinear growth in clusters. Secondly, clusters have a strong aspect of temporal coherence. While both aspects can be addressed by the Recurrent Chinese Restaurant Process (Ahmed and Xing 2008), clustering falls short of a third requirement: the model accuracy does not improve in a meaningful way as we obtain more data — doubling the time span covered by the documents simply doubles the number of clusters. But it contributes nothing to improving our language model.

Topic Models excel at the converse: They provide insight into the *content* of documents by exploiting exchangeability rather than independence when modeling documents (Blei et al. 2003). This leads to highly intuitive and human understandable document representations, yet they are

Preliminary work. Under review by AISTATS 2011. Do not distribute.

not particularly well-suited to clustering and grouping documents. For instance, they would not be capable of distinguishing between the affairs of two *different* athletes, provided that they play related sports, even if the *dramatis personae* were different. We address this challenge by building a *hierarchical* Bayesian model which contains topics at its top level and clusters drawn from a Recurrent Chinese Restaurant Process at its bottom level. In this sense it is related to Pachinko Allocation (Li and McCallum 2006) and the Hierarchical Dirichlet Process (Teh et al. 2006). One of the main differences to these models is that we mix *different datatypes*, i.e. distributions and clusters. This allows us to combine the strengths of both methods: as we obtain more documents, topics will allow us to obtain a more accurate representation of the data stream. At the same time, clusters will provide us with an accurate representation of related news articles.

A key aspect to estimation in graphical models is scalability, in particular when one is concerned with news documents arriving at a rate in excess of 1 document per second (considerably higher rates apply for blog posts). There has been previous work on scalable inference, starting with the collapsed sampler representation for LDA (Griffiths and Steyvers 2004), efficient sampling algorithms that exploit sparsity (Yao et al. 2009), distributed implementations (Smola and Narayanamurthy 2010, Asuncion et al. 2008), and Sequential Monte Carlo (SMC) estimation (Canini et al. 2009). The problem of efficient inference is exacerbated in our case since we need to obtain an *online* estimate, that is, we need to be able to generate clusters essentially on the fly as news arrives and to update topics accordingly. We address this by designing an SMC sampler which is executed in parallel by allocating particles to cores. The datastructure is a variant of the tree described by (Canini et al. 2009). Our experiments demonstrate both the scalability and accuracy of our approach when compared to editorially curated data of a major Internet news portal.

2 STATISTICAL MODEL

In a nutshell, our model emulates the process of news articles. We assume that stories occur with an approximately even probability over time. A story is characterized by a mixture of topics and the names of the key entities involved in it. Any article discussing this story then draws its words from the topic mixture associated with the story, the associated named entities, and any story-specific words that are not well explained by the topic mixture. The associated named entities and story-specific words allow the model to capture burstiness effect inside each story (Doyle and Elkan 2009, Chemudugunta et al. 2006). In summary, we model news story clustering by applying a topic model to the clusters, while simultaneously allowing for cluster generation using the Recurrent Chinese Restaurant Process (RCRP).

Such a model has a number of advantages: estimates in topic models increase with the amount of data available, hence twice as much data will lead to correspondingly improved topics. Modeling a story by its mixture of topics ensures that we have a plausible cluster model right from the start, even after observing only one article for a new story. Third, the RCRP ensures a continuous flow of new stories over time. Finally, a distinct named entity and specific-words model ensure that we capture the characteristic terms rapidly inside each story, and at the same time ensures that topics are uncorrupted by more ephemeral terms appearing in the stories (see Figure 3 for an example).

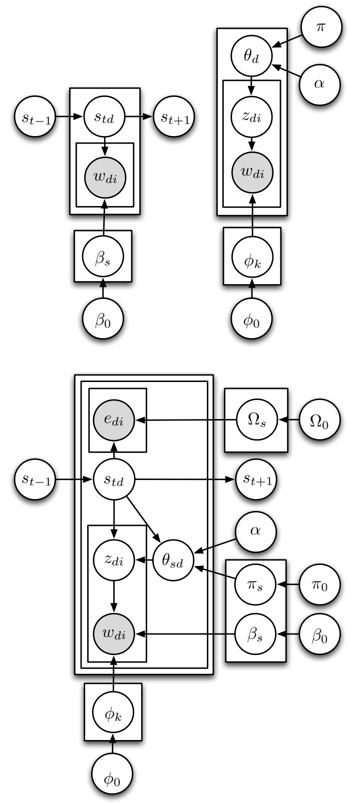

Figure 1: Plate diagram of the models. Top left: Recurrent Chinese Restaurant Process clustering; Top right: Latent Dirichlet Allocation; Bottom: Topic-Cluster model.

2.1 Recurrent Chinese Restaurant Process

A critical feature for disambiguating storylines is time. Stories come and go, and it makes little sense to try to associate a document with a storyline that has not been seen over a long period of time. We turn to the Recurrent Chinese Restaurant Process (Ahmed and Xing 2008), which generalizes the well-known Chinese Restaurant Process (CRP) (Pitman 1995) to model partially exchangeable data like document streams. The RCRP provides a nonparametric model over storyline strength, and permits sampling-based inference over a potentially unbounded number of stories.

For concreteness, we need to introduce some notation: we denote time (epoch) by t, documents by d, and the position of a word w_{di} in a document d by i. The story associated with document d is denoted by s_d (or s_{dt} if we want to make the dependence on the epoch t explicit). Documents are assumed to be divided into epochs (e.g., one hour or one day); we assume exchangeability only within each epoch. For a new document at epoch t, a probability mass proportional to γ is reserved for generating a new storyline. Each existing storyline may be selected with probability proportional to the sum $m_{st} + m'_{st}$, where m_{st} is the number of documents at epoch t that belong to storyline s, and m'_{st} is the prior weight for storyline s at time t. Finally, we denote by β_s the word distribution for story s and we let β_0 be the prior for word distributions. We compactly write

$$s_{td}|\mathbf{s_{1:t-1}}, \mathbf{s_{t,1:d-1}} \sim \mathrm{RCRP}(\gamma, \lambda, \Delta) \tag{1}$$

to indicate the distribution

$$P(s_{td}|\mathbf{s_{1:t-1}}, \mathbf{s_{t,1:d-1}}) \propto \begin{cases} m'_{st} + m^{-td}_{st} & \text{existing story} \\ \gamma & \text{new story} \end{cases} \tag{2}$$

As in the original CRP, the count m^{-td}_{ts} is the number of documents in storyline s at epoch t, not including d. The temporal aspect of the model is introduced via the prior $m^{'}_{st}$, which is defined as

$$m^{'}_{st} = \sum_{\delta=1}^{\Delta} e^{-\frac{\delta}{\lambda}} m_{s,t-\delta}. \tag{3}$$

This prior defines a time-decaying kernel, parametrized by Δ (width) and λ (decay factor). When $\Delta = 0$ the RCRP degenerates to a set of independent Chinese Restaurant Processes at each epoch; when $\Delta = T$ and $\lambda = \infty$ we obtain a global CRP that ignores time. In between, the values of these two parameters affect the expected life span of a given component, such that the lifespan of each storyline follows a power law distribution (Ahmed and Xing 2008). The graphical model is given on the top left in Figure 1.

We note that dividing documents into epochs allows for the cluster strength at time t to be efficiently computed, in terms of the components (m, m') in (2). Alternatively, one could define a continuous, time-decaying kernel over the time stamps of the documents. When processing document d at time t' however, computing any story's strength would now require summation over all earlier documents associated with that story, which is non-scalable. In the news domain, taking epochs to be one day long means that the recency of a given story decays only at epoch boundaries, and is captured by m'. A finer epoch resolution and a wider Δ can be used without affecting computational efficiency – it is easy to derive an iterative update $m'_{s,t+1} = \exp^{-1/\lambda}(m_{st} + m'_{st}) - \exp^{-(\Delta+1)/\lambda} m_{s,t-(\Delta+1)}$, which has constant runtime w.r.t. Δ.

2.2 Topic Models

The second component of the topic-cluster model is given by Latent Dirichlet Allocation (Blei et al. 2003), as described in the top right of Figure 1. Rather than assuming that documents belong to clusters, we assume that there exists a topic distribution θ_d for document d and that each word w_{di} is drawn from the distribution ϕ_t associated with topic z_{di}. Here ϕ_0 denotes the Dirichlet prior over word distributions. Finally, θ_d is drawn from a Dirichlet distribution with mean π and precision α.

1. For all topics t draw
   (a) word distribution ϕ_k from word prior ϕ_0
2. For each document d draw
   (a) topic distribution θ_d from Dirichlet prior (π, α)
   (b) For each position (d, i) in d draw
      i. topic z_{di} from topic distribution θ_d
      ii. word w_{di} from word distribution $\phi_{z_{di}}$

The key difference to the basic clustering model is that our estimate will improve as we receive more data.

2.3 Time-Dependent Topic-Cluster Model

We now combine clustering and topic models into our proposed storylines model by imbuing each storyline with a Dirichlet distribution over topic strength vectors with parameters (π, α). For each article in a storyline the topic proportions θ_d are drawn from this Dirichlet distribution – this allows documents associated with the same story to emphasize various topical aspects of the story with different degrees.

Words are drawn either from the storyline or one of the topics. This is modeled by adding an element $K + 1$ to the topic proportions θ_d. If the latent topic indicator $z_n \leq K$, then the word is drawn from the topic ϕ_{z_n}; otherwise it is drawn from a distribution linked to the storyline β_s. This story-specific distribution captures the burstiness of the characteristic words in each story.

Topic models usually focus on individual words, but news stories often center around specific people and locations For this reason, we extract named entities e_{di} from text in a preprocessing step, and model their generation directly. Note that we make no effort to resolve names "Barack Obama" and "President Obama" to a single underlying semantic entity, but we do treat these expressions as single tokens in a vocabulary over names.

1. For each topic $k \in 1 \ldots K$, draw a distribution over words $\phi_k \sim \mathrm{Dir}(\phi_0)$
2. For each document $d \in \{1, \cdots, D_t\}$:
   (a) Draw the storyline indicator $s_{td}|\mathbf{s_{1:t-1}}, \mathbf{s_{t,1:d-1}} \sim RCRP(\gamma, \lambda, \Delta)$
   (b) If s_{td} is a new storyline,
      i. Draw a distribution over words $\beta_{s_{\mathrm{new}}}|G_0 \sim \mathrm{Dir}(\beta_0)$

ii. Draw a distribution over named entities $\Omega_{s_{\mathrm{new}}}|G_0 \sim \mathrm{Dir}(\Omega_0)$
iii. Draw a Dirichlet distribution over topic proportions $\pi_{s_{\mathrm{new}}}|G_0 \sim \mathrm{Dir}(\pi_0)$

(c) Draw the topic proportions $\theta_{td}|s_{td} \sim \mathrm{Dir}(\alpha\pi_{s_{td}})$
(d) Draw the words $\mathbf{w}_{td}|s_{td} \sim \mathrm{LDA}\left(\theta_{s_{td}}, \{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}\right)$
(e) Draw the named entities $\mathbf{e}_{td}|s_{td} \sim \mathrm{Mult}(\Omega_{s_{td}})$

where $\mathrm{LDA}\left(\theta_{s_{td}}, \{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}\right)$ indicates a probability distribution over word vectors in the form of a Latent Dirichlet Allocation model (Blei et al. 2003) with topic proportions $\theta_{s_{td}}$ and topics $\{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}$.

3 INFERENCE

Our goal is to compute *online* the posterior distribution $P(\mathbf{z}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T})$, where $\mathbf{x}_t, \mathbf{z}_t, \mathbf{s}_t$ are shorthands for all of the documents at epoch t ($\mathbf{x}_{td} = \langle \mathbf{w}_{td}, \mathbf{e}_{td}\rangle$), the topic indicators at epoch t and story indicators at epoch t. Markov Chain Monte Carlo (MCMC) methods which are widely used to compute this posterior are inherently batch methods and do not scale well to the amount of data we consider. Furthermore they are unsuitable for streaming data.

3.1 Sequential Monte Carlo

Instead, we apply a sequential Monte Carlo (SMC) method known as particle filters (Doucet et al. 2001). Particle filters approximate the posterior distribution over the latent variables up until document $t, d-1$, i.e. $P(\mathbf{z}_{1:t,d-1}, \mathbf{s}_{1:t,d-1}|\mathbf{x}_{1:t,d-1})$, where $(1:t,d)$ is a shorthand for all documents up to document d at time t. When a new document td arrives, the posterior is updated yielding $P(\mathbf{z}_{1:td}, \mathbf{s}_{1:td}|\mathbf{x}_{1:td})$. The posterior approximation is maintained as a set of weighted *particles* that each represent a hypothesis about the hidden variables; the weight of each particle represents how well the hypothesis maintained by the particle explains the data.

The structure is described in Algorithms 1 and 2. The algorithm processes **one document at a time** in the order of arrival. This should not be confused with the time stamp of the document. For example, we can chose the epoch length to be a full day but still process documents inside the same day as they arrive (although they all have the same timestamp). The main ingredient for designing a particle filter is the proposal distribution $Q(\mathbf{z}_{td}, s_{td}|\mathbf{z}_{1:t,d-1}, \mathbf{s}_{1:t,d-1}, \mathbf{x}_{1:td})$. Usually this proposal is taken to be the prior distribution $P(\mathbf{z}_{td}, s_{td}|\mathbf{z}_{1:t,d-1}, \mathbf{s}_{1:t,d-1})$ since computing the posterior is hard. We take Q to be the posterior $P(\mathbf{z}_{td}, s_{td}|\mathbf{z}_{1:t,d-1}, \mathbf{s}_{1:t,d-1}, \mathbf{x}_{1:td})$, which minimizes the variance of the resulting particle weights (Doucet et al. 2001). Unfortunately computing this posterior for a single document is intractable, thus we use MCMC and run a Markov chain over $(\mathbf{z}_{td}, s_{td})$ whose equilibrium distribution is the sought-after posterior. The exact sampling equations af s and $\mathbf{z}_{td}$ are given below. This idea was inspired by the work of (Jain and Neal 2000) who used a restricted Gibbs scan over a set of coupled variables to define a proposal distribution, where the proposed value of the variables is taken to be the last sample. Jain and Neal used this idea in the context of an MCMC sampler, here we use it in the context of a sequential importance sampler (i.e. SMC).

Algorithm 1 A Particle Filter Algorithm

```
Initialize ω_1^f to 1/F for all f ∈ {1, ... F}
for each document d with time stamp t do
  for f ∈ {1, ... F} do
    Sample s_td^f, z_td^f using MCMC
    ω^f ← ω^f P(x_td | z_td^f, s_td^f, x_{1:t,d-1})
  end for
  Normalize particle weights
  if ||ω_t||_2^{-2} < threshold then
    resample particles
    for f ∈ {1, ... F} do
      MCMC pass over 10 random past documents
    end for
  end if
end for
```

Algorithm 2 MCMC over document td

```
q(s) = P(s | s_{t-Δ:t}^{-td}) P(e_td | s, rest)
for iter = 0 to MAXITER do
  for each word w_tdi do
    Sample z_tdi using (4)
  end for
  if iter = 1 then
    Sample s_td using (5)
  else
    Sample s* using q(s)
    r = P(z_td | s*, rest) P(w_td^{K+1} | s*, rest) / P(z_td | s_td, rest) P(w_td^{K+1} | s_td, rest)
    Accept s_td ← s* with probability min(r, 1)
  end if
end for
Return z_td, s_td
```

Sampling topic indicators: For the topic of word i in document d and epoch t, we sample from

$$P(z_{tdi} = k|w_{tdi} = w, s_{td} = s, \mathrm{rest}) \tag{4}$$

$$= \frac{C_{tdk}^{-i} + \alpha \frac{C_{sk}^{-i} + \pi_0}{C_{s.}^{-i} + \pi_0(K+1)}}{C_{td.}^{-i} + \alpha} \frac{C_{kw}^{-i} + \phi_0}{C_{k.}^{-i} + \phi_0 W}$$

where `rest` denotes all other hidden variables, C_{tdk}^{-i} refers to the count of topic k and document d in epoch t, not including the currently sampled index i; C_{sk}^{-i} is the count of topic k with story s, while C_{kw}^{-i} is the count of word w

with topic k (which indexes the story if $k = K + 1$); traditional dot notation is used to indicate sums over indices (e.g. $C_{td.}^{-i} = \sum_k C_{tdk}^{-i}$). Note that this is just the standard sampling equation for LDA except that the prior over the document's topic vector θ is replaced by it's story mean topic vector.

Sampling story indicators: The sampling equation for the storyline s_{td} decomposes as follows:

$$P(s_{td}|\mathbf{s}_{t-\Delta:t}^{-td}, \mathbf{z}_{td}, \mathbf{e}_{td}, \mathbf{w}_{td}^{K+1}, \text{rest}) \propto \underbrace{P(s_{td}|\mathbf{s}_{t-\Delta:t}^{-td})}_{\text{Prior}}$$
$$\times \underbrace{P(\mathbf{z}_{td}|s_{td}, \text{rest})P(\mathbf{e}_{td}|s_{td}, \text{rest})P(\mathbf{w}_{td}^{K+1}|s_{td}, \text{rest})}_{\text{Emission}} \quad (5)$$

where the prior follows from the RCRP (2), $\mathbf{w}_{td}^{K+1}$ are the set of words in document d sampled from the story specific language model $\beta_{s_{td}}$, and the emission terms for $\mathbf{w}_{td}^{K+1}, \mathbf{e}_{td}$ are simple ratios of partition functions. For example, the emission term for entities, $P(\mathbf{e}_{td}|s_{td} = s, \text{rest})$ is given by:

$$\frac{\Gamma\Big(\sum_{e=1}^{E}[C_{se}^{-td} + \Omega_0]\Big)}{\Gamma\Big(\sum_{e=1}^{E}[C_{td,e} + C_{se}^{-td} + \Omega_0]\Big)} \prod_{e=1}^{E} \frac{\Gamma\Big(C_{td,e} + C_{se}^{-td} + \Omega_0\Big)}{\Gamma\Big(C_{se}^{-td} + \Omega_0\Big)} \quad (6)$$

Since we integrated out θ, the emission term over $\mathbf{z}_{td}$ does not have a closed form solution and is computed using the chain rule as follows:

$$P(\mathbf{z}_{td}|s_{td} = s, \text{rest}) = \prod_{i=1}^{n_{td}} P(z_{tdi}|s_{td} = s, \mathbf{z}_{td}^{-td,(n\geq i)}, \text{rest}) \quad (7)$$

where the superscript $-td, (n \geq i)$ means excluding all words in document td after, and including, position i. Terms in the product are computed using (4).

We alternate between sampling (4) and (5) for 20 iterations. Unfortunately, even then the chain is too slow for online inference, because of (7) which scales linearly with the number of words in the document. In addition we need to compute this term for every active story. To solve this we use a proposal distribution

$$q(s) = P(s_{td}|\mathbf{s}_{t-\Delta:t}^{-td})P(\mathbf{e}_{td}|s_{td}, \text{rest})$$

whose computation scales linearly with the number of entities in the document. We then sample $s*$ from this proposal and compute the acceptance ratio r which is simply

$$r = \frac{P(\mathbf{z}_{td}|s*, \text{rest})P(\mathbf{w}_{td}^{K+1}|s*, \text{rest})}{P(\mathbf{z}_{td}|s_{td}, \text{rest})P(\mathbf{w}_{td}^{K+1}|s_{td}, \text{rest})}.$$

Thus we need only to compute (7) twice per MCMC iteration. Another attractive property of the proposal distribution $q(s)$ is that the proposal is *constant* and does not depend on $\mathbf{z}_{td}$. As made explicit in Algorithm 2 we precompute it once for the entire MCMC sweep. Finally, the unnormalized importance weight for particle f at epoch t, ω_t^f, can be shown to be equal to:

$$\omega^f \leftarrow \omega^f P(\mathbf{x}_{td}|\mathbf{z}_{td}^f, s_{td}^f, \mathbf{x}_{1:t,d-1}), \quad (8)$$

which has the intuitive explanation that the weight for particle f is updated by multiplying the marginal probability of the new observation $\mathbf{x}_t$, which we compute from the last 10 samples of the MCMC sweep over a given document. Finally, if the effective number of particles $\|\omega_t\|_2^{-2}$ falls below a threshold we stochastically replicate each particle based on its normalized weight. To encourage diversity in those replicated particles, we select a small number of documents (10 in our implementation) from the recent 1000 documents, and do a single MCMC sweep over them, and then finally reset the weight of each particle to uniform.

We note that an *alternative approach* to conducting the particle filter algorithm would sequentially order s_{td} followed by $\mathbf{z}_{td}$. Specifically, we would use $q(s)$ defined above as the proposal distribution over s_{td}, and then sample $\mathbf{z}_{td}$ sequentially using Eq (4) conditioned on the sampled value of s_{td}. However, this approach requires a huge number of particles to capture the uncertainty introduced by sampling s_{td} before actually seeing the document, since $\mathbf{z}_{td}$ and s_{td} are tightly coupled. Moreover, our approach results in less variance over the posterior of $(\mathbf{z}_{td}, s_{td})$ and thus requires fewer particles, as we will demonstrate empirically.

3.2 Speeding up the Sampler

While the sampler specified by (4) and (5) and the proposal distribution $q(s)$ are efficient, they scale linearly with the number of topics and stories. Yao et al. (2009) noted that samplers that follow (4) can be made more efficient by taking advantage of the sparsity structure of the word-topic and document-topic counts: each word is assigned to only a few topics and each document (story) addresses only a few topics. We leverage this insight here and present an extended, efficient data structure in Section 3.3 that is suitable for particle filtering.

We first note that (4) follows the standard form of a collapsed Gibbs sampler for LDA, albeit with a story-specific prior over θ_{td}. We make the approximation that the document's story-specific prior is constant while we sample the document, i.e. the counts C_{sk}^{-i} are constants. This turns the problem into the same form addressed in (Yao et al. 2009). The mass of the sampler in (4) can be broken down into three parts: prior mass, document-topic mass and word-topic mass. The first is dense and constant (due to our approximation), while the last two masses are sparse. The document-topic mass tracks the non-zero terms in C_{tdk}^{-i}, and the word-topic mass tracks the non-zero terms in C_{kw}^{-i}.

The sum of each of these masses can be computed once at the beginning of the sampler. The document-topic mass can be updated in $O(1)$ after each word (Yao et al. 2009), while the word-topic mass is very sparse and can be computed for each word in nearly constant time. Finally the prior mass is only re-computed when the document's story changes. Thus the cost of sampling a given word is almost *constant* rather than $O(k)$ during the execution of Algorithm 1.

Unfortunately, the same idea can not be applied to sampling s, as each of the components in (5) depends on *multiple* terms (see for example (6)). Their products do not fold into separate masses as in (4). Still, we note that the entity-story counts are sparse ($C_{se}^{-td} = 0$), thus most of the terms in the product component ($e \in E$) of (6) reduce to the form $\Gamma(C_{td,e} + \Omega_0)/\Gamma(\Omega_0)$. Hence we simply compute this form once for all stories with $C_{se}^{-td} = 0$; for the *few* stories having $C_{se}^{-td} > 0$, we explicitly compute the product component. We also use the same idea for computing $P(\mathbf{w}_{td}^{K+1}|s_{td}, \text{rest})$. With these choices, the entire MCMC sweep for a given document takes around *50-100ms* when using `MAXITER = 15` and $K = 100$ as opposed to *200-300ms* for a naive implementation.

Hyperparameters:

The hyperparameters for topic, word and entity distributionss, ϕ_0, Ω_0 and β_0 are *optimized* as described in Wallach et al. (2009) every 200 documents. The topic-prior $\pi_{0,1:K+1}$ is modeled as asymmetric Dirichlet prior and is also *optimized* as in Wallach et al. (2009) every 200 documents. For the RCRP, the hyperparameter γ_t is epoch-specific with a Gamma(1,1) prior; we *sample* its value after every batch of 20 documents (Escobar and West 1995). The kernel parameters are set to $\Delta = 3$ and $\lambda = 0.5$ — results were robust across a range of settings. Finally we set $\alpha = 1$

3.3 Implementation and Storage

Implementing our parallel SMC algorithm for large datasets poses memory challenges. Since our implementation is multi-threaded, we require a *thread-safe* data structure supporting fast updates of individual particles' data, *and* fast copying of particle during re-sampling step. We employ an idea from 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. To ensure *thread safety*, we augment the inheritance tree by requiring every particle to have its own tree *leaf*. This makes particle writes thread-safe, since no particle is ever an ancestor of another (see Appendix for details).

Extended inheritance trees Parts of our algorithm require storage of *sets of objects*. For example, our story sampling equation (5) needs the set of stories associated with each named entity, as well as the number of times each story-to-entity association occurs. To solve this problem, we extend the basic inheritance tree by making its

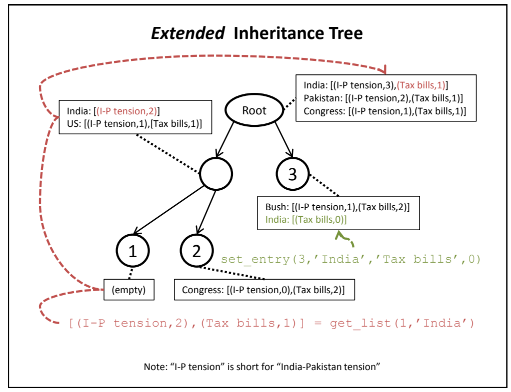

Figure 2: Operations on an *extended inheritance tree*, which stores sets of objects in particles, shown as lists in tables connected to particle-numbered tree nodes. Our algorithm requires particles to store some data as sets of objects instead of arrays — in this example, for every named entity, e.g. "Congress", we need to store a set of (story,association-count) pairs, e.g. ("Tax bills",2). The extended inheritance tree allows (a) the particles to be replicated in constant-time, and (b) the object sets to be retrieved in amortized linear time. Notice that every particle is associated with a leaf, which ensures thread safety during write operations. Internal vertices store entries common to leaf vertices.

hash maps store *other hash maps* as values. These second-level hash maps then store objects as key-value pairs; note that individual objects *can be shared with parent vertices*. Using the story sampling equation (5) 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 shows an example with stories taken from Figure 3). Observe that the count for a particular story-entity association can be retrieved or updated in amortized constant time. Retrieving all associations for a given entity is usually linear in the number of associations. Finally note that the list associated with each key (NE or word) is not sorted as in Yao et al. (2009) as this will prevent sharing across particles. Nevertheless, our implementation balances storage and execution time.

4 EXPERIMENTS

We examine our model on *three* English news samples of varying sizes extracted from Yahoo! News over a two-month period. Details of the three news samples are listed in Table 1. We use the named entity recognizer in (Zhou et al. 2010), and we remove common stop-words and tokens which are neither verbs, nor nouns, nor adjectives. We divide each of the samples into a set of 12-hour epochs (corresponding to AM and PM time of the day) according to the article publication date and time. For all experiments, we use 8-particles running on an 8-core machine, and unless otherwise stated, we set **MAXITER**=15.

4.1 Structured Browsing

In Figure 3 we present a qualitative illustration of the utility of our model for structure browsing. The storylines include the UEFA soccer championships, a tax bill under consideration in the United States, and tension between

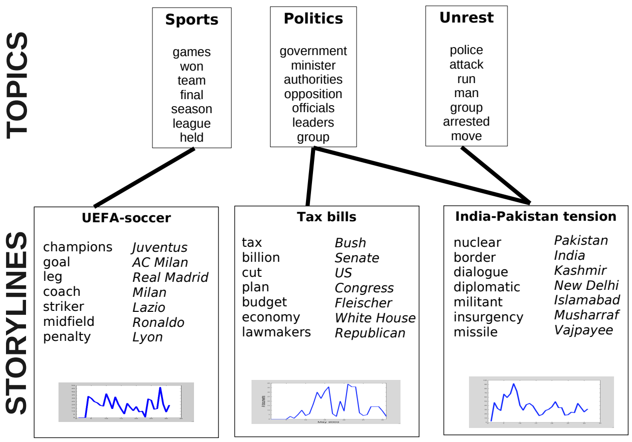

Figure 3: Some example storylines and topics extracted by our system. For each storyline we list the top words in the left column, and the top named entities at the right; the plot at the bottom shows the storyline strength over time. For topics we show the top words. The lines between storylines and topics indicate that at least 10% of terms in a storyline are generated from the linked topic.

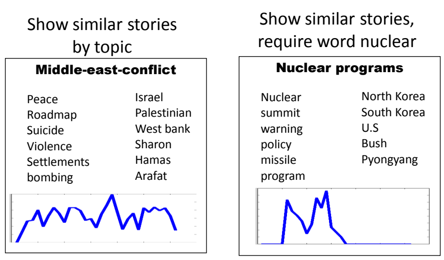

Figure 4: An example of structure browsing of documents related to the India-Pakistan tensions (see text for details).

India and Pakistan. Our model identifies connections between these storylines and relevant high-level topics: the UEFA story relates to a more general topic about sports; both the tax bill and the India-Pakistan stories relate to the politics topics, but only the latter story relates to the topic about civil unrest. Note that each storyline contains a plot of strength over time; the UEFA storyline is strongly multimodal, peaking near the dates of matches. This demonstrates the importance of a flexible nonparametric model for time, rather than using a unimodal distribution.

End-users can take advantage of the organization obtained by our model, by browsing the collection of high-level topics and then descending to specific stories indexed under each topic. In addition, our model provides a number of affordances for structured browsing which were not possible under previous approaches. Figure 4 shows two examples that are retrieved starting from the India-Pakistan tension story: one based on similarity of high-level topical content θ_s, and the other obtained by focusing the query on similar stories featuring the topic politics but requiring the keyword *nuclear* to have high salience in the term probability vector of any story returned by the query. This combination of topic-level analysis with surface-level matching on terms or entities is a unique contribution of our model, and was not possible with previous technology.

Table 1: Details of Yahoo! News dataset and corresponding clustering accuracies of the baseline (LSHC) and our method (Story), $K = 100$.

Sample No.	Sample size	Num Words	Num Entities	Story Acc.	LSHC Acc.
1	111,732	19,218	12,475	**0.8289**	0.738
2	274,969	29,604	21,797	**0.8388**	0.791
3	547,057	40,576	32,637	**0.8395**	0.800

Table 2: Clustering accuracies vs. number of topics.

sample-No.	K=50	K=100	K=200	K=300
1	0.8261	**0.8289**	0.8186	0.8122
2	0.8293	**0.8388**	0.8344	0.8301
3	**0.8401**	0.8395	0.8373	0.8275

Table 3: The effect of hyperparameters on sample-1, with $K = 100$, $\phi_0 = .01$, and no hyperparameter optimization.

	$\beta_0 = .1$	$\beta_0 = .01$	$\beta_0 = .001$
$\Omega_0 = .1$	0.7196	0.7140	0.7057
$\Omega_0 = .01$	0.7770	0.7936	0.7845
$\Omega_0 = .001$	0.8178	**0.8209**	**0.8313**

Table 4: Component Contribution, sample-1, $K = 100$.

Removed Feature	Time	Names entites	Story words	Topics *(equiv. RCRP)*
Accuracy	0.8225	.6937	0.8114	0.7321

Table 5: Number of particles, sample-1, $K = 100$.

#Particles	4	8	16	32	50
Accuracy	0.8101	08289	0.8299	0.8308	0.8358

4.2 Evaluating Clustering Accuracy

We evaluate the clustering *accuracy* of our model over the Yahoo! news datasets. Each dataset contains 2525 editorially judged "must-link" (45%) and "cannot-link" (55%) article pairs. Must-link pairs refer to articles in the same story, whereas cannot-link pairs are not related.

For the sake of evaluating clustering, we compare against a variant of a strong 1-NN (single-link clustering) baseline (Connell et al. 2004). This simple baseline is the best performing system on TDT2004 task and was shown to be competitive with Bayesian models (Zhang et al. 2004). This method finds the closest 1-NN for an incoming document among all documents seen thus far. If the distance to this 1-NN is above a threshold, the document starts a new story, otherwise it is linked to its 1-NN. Since this method examines all previously seen documents, it is not scalable to large-datasets. In (Petrovic et al. 2010), the authors showed that using locality sensitive hashing (LSH), one can restrict the subset of documents examined with little effect of the final accuracy. Here, we use a similar idea, but we even allow the baseline to be fit *offline*. First, we compute the similarities between articles via LSH (Haveliwala et al.

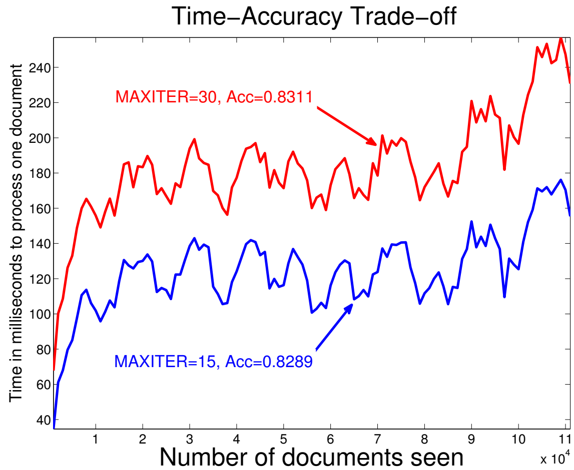

Figure 5: Effect of MAXITER, sample-1, $K = 100$

2000, Gionis et al. 1999), then construct a pairwise similarity graph on which a single-link clustering algorithm is applied to form larger clusters. The single-link algorithm is stopped when no two clusters to be merged have similarity score larger than a threshold *tuned* on a separate validation set (our algorithm has *no access* to this validation set). In the remainder of this paper, we simply refer to this baseline as LSHC.

From Table 1, we see that our *online, single-pass* method is competitive with the *off-line* and tuned baseline on all the samples and that the difference in performance is larger for small sample sizes. We believe this happens as our model can isolate story-specific words and entities from background topics and thus can link documents in the same story even when there are few documents in each story.

4.3 Hyperparameter Sensitivity

We conduct five experiments to study the effect of various model hyperparameters and tuning parameters. First, we study the effect of the number of topics. Table 2 shows how performance changes with the number of topics K. It is evident that $K = 50-100$ is sufficient. Moreover, since we optimize π_0, the effect of the number of topics is negligible (Wallach et al. 2009) For the rest of the experiments in this section, we use sample-1 with $K = 100$.

Second, we study the number of Gibbs sampling iterations used to process a single document, `MAXITER`. In Figure 5, we show how the time to process each document grows with the number of processed documents, for different values of `MAXITER`. As expected, doubling `MAXITER` increases the time needed to process a document, however performance only increases marginally.

Third, we study the effectiveness of optimizing the hyperparameters ϕ_0, β_0 and Ω_0. In this experiment, we turn off hyperparameter optimization altogether, set $\phi_0 = .01$ (which is a common value in topic models), and vary β_0 and Ω_0. The results are shown in Table 3. Moreover, when we enable hyperparameter optimization, we obtain $(\phi_0, \beta_0, \Omega_0) = (0.0204, 0.0038, 0.0025)$ with accuracy 0.8289, which demonstrates its effectiveness.

Fourth, we tested the contribution of each feature of our model (Table 4). As evident, each aspect of the model improves performance. We note here that removing time not only makes performance suboptimal, but also causes stories to persist throughout the corpus, eventually increasing running time to an astounding 2 seconds per document.

Finally, we show the effect of the number of particles in Table 5. This validates our earlier hypothesis that the restricted Gibbs scan over $(\mathbf{z}_{td}, s_{td})$ results in a posterior with small variance, thus only a few particles are sufficient to get good performance.

5 RELATED WORK

Our problem is related to work done in the topic detection and tracking community (TDT), however the latter work focuses on clustering documents into stories, mostly by way of surface level similarity techniques and single-link clustering (Connell et al. 2004). Moreover, there is little work on obtaining two-level organizations (e.g. Figure 3) in an *unsupervised* and *data-driven* fashion, nor in summarizing each story using general topics in addition to specific words and entities – thus our work is unique in this aspect.

Our approach is non-parametric over stories, allowing the number of stories to be determined by the data. In similar fashion Zhang et al. (2004) describe an online clustering approach using the Dirichlet Process. This work equates storylines with clusters, and does not model high-level topics. Also, non-parametric clustering has been previously combined with topic models, with the cluster defining a distribution over topics (Yu et al. 2005, Wallach 2008). We differ from these approaches in several respects: we incorporate temporal information and named entities, and we permit both the storylines and topics to emit words.

Recent work on topic models has focused on improving scalability; we focus on sampling-based methods, which are most relevant to our approach. Our approach is most influenced by the particle filter of Canini et al. (2009), but we differ in that the high-order dependencies of our model require special handling, as well as an adaptation of the sparse sampler of Yao et al. (2009).

6 CONCLUSIONS

We present a scalable probabilistic model for extracting storylines in news and blogs. The key aspects of our model are (1) a principled distinction between topics and storylines, (2) a non-parametric model of storyline strength over time, and (3) an online efficient inference algorithm over a non-trivial dynamic non-parametric model. We contribute a very efficient data structure for fast-parallel sampling and demonstrated the efficacy of our approach on hundreds of thousands of articles from a major news portal.

Acknowledgments

We thank the anonymous reviewers for their helpful comment, and Yahoo! Research for the datasets. This work is supported in part by grants NSF IIS- 0713379, NSF DBI-0546594 career award, ONR N000140910758, DARPA NBCH1080007, AFOSR FA9550010247, and Alfred P. Sloan Research Fellowsh to EPX.

References

Ahmed, A. and E. P. Xing (2008). Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *SDM*, pp. 219–230. SIAM.

Antoniak, C. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics 2*, 1152–1174.

Asuncion, A., P. Smyth, and M. Welling (2008). Asynchronous distributed learning of topic models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.), *NIPS*, pp. 81–88. MIT Press.

Blei, D., A. Ng, and M. Jordan (2003, January). Latent Dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

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)*.

Chemudugunta, C., P. Smyth, and M. Steyvers (2006). Modeling general and specific aspects of documents with a probabilistic topic model. In *NIPS*.

Connell, M., A. Feng, G. Kumaran, H. Raghavan, C. Shah, and J. Allan (2004). Umass at tdt 2004. In *TDT 2004 Workshop Proceedings*.

Doucet, A., N. de Freitas, and N. Gordon (2001). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Doyle, G. and C. Elkan (2009). Accounting for burstiness in topic models. In *ICML*.

Escobar, M. and M. West (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association 90*, 577–588.

Gionis, A., P. Indyk, and R. Motwani (1999). Similarity search in high dimensions via hashing. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie (Eds.), *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, pp. 518–529. Morgan Kaufmann.

Griffiths, T. and M. Steyvers (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences 101*, 5228–5235.

Haveliwala, T., A. Gionis, and P.Indyk. (2000). Scalable techniques for clustering the web. In *WebDB*.

Jain, S. and R. Neal (2000). A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics 13*, 158–182.

Li, W. and A. McCallum (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. *ICML*.

Petrovic, S., M. Osborne, and V. Lavrenko (2010). Streaming first story detection with application to twitter. In *NAACL*.

Pitman, J. (1995). Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields 102*(2), 145–158.

Smola, A. and S. Narayanamurthy (2010). An architecture for parallel topic models. In *Very Large Databases (VLDB)*.

Teh, Y., M. Jordan, M. Beal, and D. Blei (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association 101*(576), 1566–1581.

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

Wallach, H. M., D. Mimno, and A. McCallum. (2009). Rethinking lda: Why priors matter. In *NIPS*.

Yao, L., D. Mimno, and A. McCallum (2009). Efficient methods for topic model inference on streaming document collections. In *KDD'09*.

Yu, K., S. Yu, , and V. Tresp (2005). Dirichlet enhanced latent semantic analysis. In *AISTATS*.

Zhang, J., Y. Yang, and Z. Ghahramani (2004). A probabilistic model for online document clustering with application to novelty detection. In *Neural Information Processing Systems*.

Zhou, Y., L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney (2010, August). Resolving surface forms to wikipedia topics. In *Proceedings of the 23rd International Conference on Computational Linguistics COLING*, pp. 1335–1343.