Unified Analysis of Streaming News

Amr Ahmed, Qirong Ho
Jacob Eisenstein, Eric P. Xing
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{amahmed,qho+,jacobeis,epxing}@cs.cmu.edu

Alexander J. Smola, Choon Hui Teo
Yahoo! Research
Santa Clara, CA 95051, USA
{smola, choonhui}@yahoo-inc.com

ABSTRACT

News clustering, categorization and analysis are key components of any news portal. They require algorithms capable of dealing with dynamic data to cluster, interpret and to temporally aggregate news articles. These three tasks are often solved separately. In this paper we present a unified framework to group incoming news articles into temporary but tightly-focused *storylines*, to identify prevalent topics and key entities within these stories, and to reveal the temporal structure of stories as they evolve. We achieve this by building a hybrid clustering and topic model. To deal with the available wealth of data we build an efficient parallel inference algorithm by sequential Monte Carlo estimation. Time and memory costs are nearly constant in the length of the history, and the approach scales to hundreds of thousands of documents. We demonstrate the efficiency and accuracy on the publicly available TDT dataset and data of a major internet news site.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligenc**]: Learning; G.3 [**Probability And Statistics**]:

General Terms

Algorithms, Experimentation

Keywords

Topic Models, Dirichlet Processes, Online Inference

1. INTRODUCTION

Internet news portals provide an increasingly important service for information dissemination.To ensure good performance they need to provide a number of 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 take previous news into account. This is particularly important for stories that are just about to drop off the radar — a single article on the Haiti earthquake a year after the event may not carry much statistical weight, yet when combined with past information it may be categorized efficiently into the bigger context of related news.

Mindshare: When viewing events it is useful to gauge their importance in terms of whether the story is currently "trendy", and to assess how popular it has been in the past. Services such as Google Trends or Yahoo! Buzz provide similar functionality.

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 rock 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 will simply increase the number of clusters, without necessarily improving their quality — but it holds for topic models. Yet topic models are unsuitable for direct analysis since they do not reason well at the level of individual events.

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.

Our approach relies on a hybrid between clustering algorithms and topic models. We use clustering to address the issue of aggregating related news into the same story, focusing on the "who," "when," and "where" of the story. At the same time, topic models allow us to identify the "what" and "how," tying new stories to related events from the past. While topic models and clustering are usually represented as opposing approaches in the literature, we show here that they can be combined into a powerful hierarchical framework.

Furthermore, we use time-dependent dynamic cluster assumptions to make our model adaptive to the ever changing nature of the news stream. At its heart is a nonparametric Bayesian approach called the Recurrent Chinese Restaurant Process [3]. As a side benefit our model yields temporal intensity tracking, i.e. it addresses the issue of constructing

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

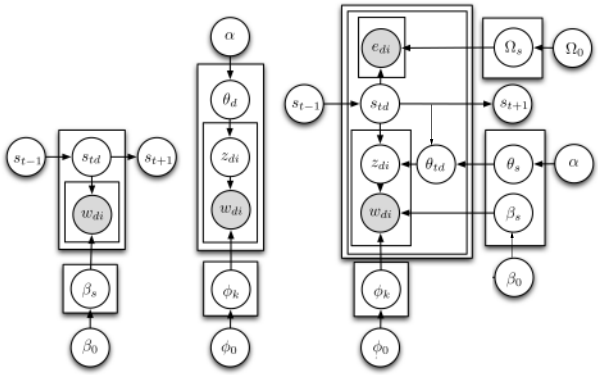

Figure 1: Left: Recurrent Chinese Restaurant Process for clustering; Middle: Latent Dirichlet Allocation; Right: Storylines.

timelines of news.

A key departure from prior work is our emphasis on computationally scalable inference. Our ultimate goal is an online system that can run continuously, with time and memory costs that are constant in the length of the history. To this end, we develop a stochastic Monte Carlo inference procedure that collapses out many of the model's latent variables, yet remains trivially parallelizable.

Our experiments demonstrate both the scalability and accuracy of our approach. In particular, we compare against the manual annotations from the Topic Detection and Tracking shared task [5], obtaining state of the art results (without using any of the hand annotations typically employed for parameter tuning on this task) that validate the design choices embodied by our model. To demonstrate scalability, we apply our approach to the commercial news stream of a large internet news site.

Our experiments demonstrate both the scalability and accuracy of our approach. In particular, we compare against the manual annotations from the Topic Detection and Tracking shared task [5], and editorially-labeled clusters from commercial news stream of a large internet news site.

Outline: We begin with a high-level overview of the problem of news clustering and analysis. This is followed by detailed yet relatively equation free description of the statistical approach we take. This section is sufficient to gain the intuition needed for understanding the inner workings of the model. Subsequently, we provide a more thorough mathematical presentation of the steps required in obtaining such an estimate. A discussion of the data types required for inference and implementation details follows. We conclude with our experimental results and a discussion of future work. For an in-depth analysis of the inference procedure, more detailed statistical and technical implications, we refer the reader to [1, 2].

2. CONCEPTS

Generative statistical models enable rich inference over an intuitively appealing probabilistic representation for documents and text. But just as human readers are overwhelmed by the number of potentially relevant documents, existing generative models can be deployed only on narrowly-selected collections. Standard Bayesian inference does not scale easily to web-size data [31, 12]; however, we believe that the scalability of topic models such as Latent Dirichlet Allocation is also limited on a more fundamental level by the underlying *representation*. We address both issues: we develop a novel multiresolution model that represents streaming text and we show how inference can be made efficient.

The standard topic model representation is flat, applying a single set of topics across all documents. But in blogs and news articles, there are at least two relevant dimensions, which topic models conflate. High-level topics – such as "basketball" or "financial crises" – each pertain to many distinct *storylines*, such as the bankruptcy of Lehman Brothers in 2008. The distinction between topics and storylines is not simply a question of granularity: topics feature a loose and shifting cast of characters over a long timescale, while storylines tend to occur over a compact timespan and focus on a few key actors and events. Within a given storyline, an author may choose to emphasize a small subset of all potentially relevant topics; for example, an article describing the Lehman Brothers collapse might focus on either the political or the economic aspects of the story.

We present a model capable of reasoning about an unbounded number of storylines, in relation to higher-level topics, named entities (e.g. people and locations), and time. The strength of each individual storyline rises and falls in a non-parametric model based on the Recurrent Chinese Restaurant Process [3]. The model distinguishes between storylines and high-level topics in several other respects. First, while storylines and topics both generate the words in an article, only storylines are imbued with a distribution over named entities. Second, documents are modeled as an admixture of topics, but we permit only a single latent storyline indicator which defines a prior on the topic proportions.

We assume that we have access to a continuous stream of news articles which needs to be processed instantly, and which is too large to afford periodic analysis of the *entire* collection of news articles. We assume, in particular, that we have an incoming stream of news in excess of one new article per second, and that we would like to build a system capable of processing this amount of data.

To streamline the presentation for the purpose of this paper, we ignore issues such as the identity of a news source, possible attached categorical metadata, or links present within the individual news articles. While these issues are all critical for practical deployment, they would detract from the key algorithm presented in this paper and are hence omitted. Nevertheless, it is easy to add such metadata to our model by 'upstream conditioning' [22], i.e. by using side information such as links, newspapers, authors, or categories to improve topic and cluster estimates. Similarly, co-clicks, popularity of articles, etc. can provide highly valuable information regarding the coherence of stories.

2.1 Recurrent Chinese Restaurant Process

We begin by introducing our model's time-dependent clustering component, based on the Recurrent Chinese Restaurant Process (RCRP) [3]. This is depicted in Figure 1 on the left, for which some terminology is in order: shaded variables are observed data, blank ones are latent (not observed). Arrows encode dependence, i.e. the distribution of children depends on their parents. The graphs themselves are acyclic. Plates are the statistical equivalent of a `for` loop. They refer to instantiations of the contained random variable for all values of the associated index.

The basic idea is that stories (i.e. clusters) are composed of articles and that the popularity of stories varies over time.

We assume that past story popularity is a good prior assumption for today's popularity — consequently, if we do not see articles related to a given story for a while, our prior assumption to see any further articles decreases. The RCRP not only models these assumptions, but also has the advantage of *not* assuming a fixed number of stories. Instead, it allows for a certain probability of seeing a new cluster at any given time.

Figure 1 provides a graphical depiction of this dependence. The observed words arise from a multinomial distribution, i.e. a bag of words model that is associated with the story at hand. Over time, the prevalence of stories changes, and the present article is drawn from this changing story distribution. As a result, we obtain both an estimator over the story distributions *and* a model of the stories.

However, simply clustering articles based on this model will not work: as we obtain increasingly large amounts of data by recording articles over time, our algorithm will not learn anything new from the composition of articles related to a story. In other words, if we had two years of data rather than one to build our model, we would simply obtain twice as many clusters (assuming that the average arrival rate of new stories is constant) but no further information regarding the language model underlying this process.

2.2 Topics

In order to address the clustering issue, we shall integrate our RCRP story model with the strengths of the Latent Dirichlet Allocation model [11]. The LDA exploits long-range relations between words by assuming that documents are composed of topics. Under LDA, to model the content of a document it suffices to estimate the topics that occur in it. A graphical description of LDA is given in the middle diagram of Figure 1.

The generative process first draws a topic mixture over topics for a given document. This topic mixture corresponds to a mixture of multinomial word distributions, from which words are subsequently drawn. This means that, rather than assigning a single multinomial distribution to each cluster, we now have a *mixture of multinomials* for each document. This approach has been shown to yield content models that correlate well with human judgments [13].

Unfortunately, when applying LDA to news articles we face a problem: LDA offers us little guidance in terms of how to aggregate articles into stories. That is, while LDA is very useful in identifying the content of an article such as 'finance', 'basketball', 'middle east', etc., it fails to distinguish between, say, two *different* athletes committing adultery, since both are likely concerned with the same set of topics. Unlike clustering however, LDA can incorporate more data without dramatically increasing the size of the latent space. Moreover there are effective models for dealing with topic drift over time [10]. In other words, LDA excels in almost all aspects where clustering fails, and vice versa.

2.3 Storylines and Topics

We now describe how we integrate the strengths of story clustering and LDA topic modeling. As a new story develops, we draw from a mix of topics to describe its content, in the same fashion as LDA. However, we do *not* identify a story with a single article. Instead, we assume that articles are written based on the story. That is, we treat each story as a *cluster* from which individual articles are drawn.

Such an approach combines the strengths of clustering and topic models. We use topics to describe the content of each cluster, and then we draw articles from the associated story. This is a more natural fit for the actual process of how news is created: after an event occurs (the story), several journalists write articles addressing various aspects of the story. While their vocabulary and their view of the story may differ, they will by necessity agree on the key issues related to a story (at least in terms of their vocabulary). Hence, to analyze a stream of incoming news we need to infer a) which (possibly new) cluster could have generated the article and b) which topic mix describes the cluster best.

To make the described approach work requires some modifications. First, story prevalence has to be dynamic, changing with time — this can be addressed by the storylines model of [3]. Second, we need the clusters to gain a more detailed description as we observe more data. This is achieved by allowing for cluster-specific corrections to the topic mixture.

2.4 Named Entities

While words may be drawn from distributions corresponding to either the storyline or a high-level topic, all entities are drawn from a distribution corresponding to the storyline. While entities are obviously strongly correlated with topics, their choice is somewhat orthogonal to the story itself (e.g. the name of the dying rockstar or the name of the philandering athlete). The rationale is that high-level topics are more general than any individual entity, while storylines are characterized most distinctively by the people, places, and organizations they describe. This assumption may occasionally be violated for extremely influential entities who are ubiquitous within a high-level topic, but it provides a crucial anchor that helps us to differentiate storylines from topics, while sidestepping some of the problems encountered by similar generative models over terms and entities [24]. As an added benefit, a separate model for entities provides us with useful data for annotating clusters by capturing the *dramatis personae*.

3. STATISTICAL MODEL

3.1 Recurrent Chinese Restaurant Process

A critical feature for disambiguating storylines is time. Storylines comes and go, and it makes little sense to try to associate a document with storyline that has not been seen over a long period of time. One approach to modeling time would be parametric: we could associate a probability distribution with the timestamps of documents in any given storyline [30]. However, different stories often have radically different temporal characteristics, and multimodality is not uncommon, as we observe e.g. in Figure 4.

We turn to the Recurrent Chinese Restaurant Process [3], which generalizes the well-known Chinese Restaurant Process (CRP) [27] 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.

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.

More formally, denote by s_{td} the story index associated with document d at epoch t. We can compactly write

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

to indicate the following distribution:

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

As in the original CRP, the count m_{ts}^{-td} 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

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

This prior defines a time-decaying kernel, parametrized by Δ (width) and λ (decay factor). When Δ=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 [3]. The associated graphical model is given on the left in Figure 1. The variables are as follows:

t	time
d	document
(d, i)	position i in document d
s_d	story associated with document d
w_{di}	word i in document d
β_s	word distribution for story s
β_0	prior for word distributions

For each time period $t \in \{1, \ldots, T\}$ do

For each document d in time period t do

i. Draw the storyline indicator: s_{td} via $s_{td}|\mathbf{s_{1:t-1}}, \mathbf{s_{t,1:d-1}}$

ii. If s_{td} is a new storyline draw a distribution over words $\beta_s|\beta_0$

iii. For each i in document draw $w_{di} \sim \beta_{s_{td}}$

This time-dependent clustering process constitutes the first component of the storylines model. The second component is given by a topic model.

3.2 Topic Models

The Latent Dirichlet Allocation model [11] is described in the middle of Figure 1. We have the following variables and associated generative process:

α	Dirichlet prior over topic distributions
d	document
θ_d	topic distribution for document d
(d, i)	position i in document d
z_{di}	topic associated with word at (d, i)
w_{di}	word at (d, i)
ϕ_0	Dirichlet prior over word distributions for topics
ϕ_k	word distribution for topic k

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

The key difference to the basic clustering model is that we continue improving our estimate of the stories as we receive more data.

3.3 Storylines for News

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 α (they encode mean and precision). For each article in a given storyline, the topic proportions θ_d are drawn from this Dirichlet distribution.

Words are drawn either from the storyline or one of the topics. This can be 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 β_{z_n}; otherwise it is drawn from a distribution linked to the storyline ϕ_s.

Topic models usually focus on individual words, but news stories often center around specific people and locations [21]. For this reason, we extract named entities 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. The variables are as follows:

t	time
d	document
(t, d)	document d in epoch t
(d, i)	position i in document d (word or entity)
s_{td}	story associated with document d
s_t	aggregate cluster variables at time t
e_{di}	entity at position i in document d
z_{di}	topic at position i in document d
w_{di}	word at position i in document d
Ω_s	Entity distribution for story s
Ω_0	prior for entity distributions
θ_s	topic distribution for story s
θ_{td}	topic distribution for document d at time t
α	Dirichlet prior over topic distributions
β_s	word distribution for story specific "topic" for story s
β_0	prior for story-specific word distributions
ϕ_k	word distributions for topics k
ϕ_0	Dirichlet prior over word distributions for topics

The full generative process is stated below. The key modifications are that we need to interleave the RCRP with LDA.

For each time period t from 1 to T do (forward in time)

1. For each document $d \in \{1, \cdots, D_t\}$ in epoch t do
   (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 $\beta_{s_{\text{new}}}|G_0 \sim \text{Dir}(\beta_0)$
      ii. Draw a distribution over named entities $\Omega_{s_{\text{new}}}|G_0 \sim \text{Dir}(\Omega_0)$

iii. Draw a Distribution over topic proportions $\theta_{s_{\text{new}}} \sim \text{Dir}(\alpha)$

(c) Draw the topic proportions: $\theta_{td}|s_{td} \sim \text{Dir}(\theta_{s_{td}})$

(d) Draw the words $\mathbf{w}_{td}|s_{td} \sim \text{LDA}\left(\theta_{s_{td}}, \{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}\right)$

(e) Draw the named entities $\mathbf{e}_{td}|s_{td} \sim \text{Mult}(\Omega_{s_{td}})$

Here $\text{LDA}\left(\theta_{s_{td}}, \{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}\right)$ indicates a probability distribution over word vectors in the form of a Latent Dirichlet Allocation model [11] with topic proportions θ_{td} and topics $\{\phi_1, \cdots, \phi_K, \beta_{s_{td}}\}$. The hyperparameters $\beta_0, \Omega_0, \phi_0$ are all symmetric Dirichlet.

4. 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 applications.

4.1 Sequential Monte Carlo

Instead, we apply a sequential Monte Carlo (SMC) method known as particle filters [15]. Particle filters approximate the posterior distribution over the latent variables up until document td. When document td arrives, the posterior is updated and the posterior approximation is maintained as a set of weighted *particles* 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 of the algorithm is described in Figure 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 $Q(\mathbf{z}_{td}, s_{td}|\mathbf{x}_{td}, \text{past})$. Usually this proposal is taken to be the prior distribution $P(\mathbf{z}_t, \mathbf{s}_t|past)$ since computing the posterior is hard. We take Q to be the posterior which minimizes the variance of the resulting particle weights [15]. Unfortunately computing this posterior 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 derivation for the sampling equations of z_{td}, s_{td} is given in [1, 2].

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

$$P(z_{tdi} = k|w_{tdi} = w, s_{td} = s, \texttt{rest}) \quad (4)$$
$$= \frac{C^{-i}_{tdk} + \frac{C^{-i}_{sk}+\alpha}{C^{-i}_{s.}+\alpha(K+1)}}{C^{-i}_{td.}+1} \frac{C^{-i}_{kw}+\phi_0}{C^{-i}_{k.}+\phi_0 W}$$

where `rest` denotes all other hidden variables,C^{-i}_{tdk} refers to the count of topic k and document d in epoch t, not including the currently sampled index i; C^{-i}_{sk} is the count of topic k with story s, C^{-i}_{kw} 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. $C^{-i}_{td.} = \sum_k C^{-i}_{tdk}$). 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} on a high level decomposes as follows:

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

where the prior follows from the RCRP (2), $\mathbf{w}^{K+1}_{td}$ are the set of words in document d sampled from the story specific language model $\phi_{s_{td}}$, and the emission terms for $\mathbf{w}^{K+1}_{td}, e_{td}$ are simple ratios of partition functions. 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:

$$P(\mathbf{z}_{td}|s_{td} = s, \text{rest}) = \prod_{i=1}^{n_{td}} P(z_{tdi}|s_{td} = s, \mathbf{z}^{-td,(n\geq i)}_{td}, \text{rest}) \quad (6)$$

where the superscript $-td, (n \geq i)$ means that we exclude all words in document td that came after position i. The terms in the product can be computed using (4).

We alternate between sampling (4) and (5) for 15 iterations (as we showed in [1, 2], increasing the number of iterations beyond 15 does not help). Unfortunately, even then the chain is too slow for online inference, because of (6) 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

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

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

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

Thus we need only to compute (6) twice per MCMC iteration. Another attractive property is that the proposal is constant and does not depend on $\mathbf{z}_{td}$, thus, we precompute it once for the entire MCMC sweep. Finally, the unnormalized importance weight for particle f, ω^f after td is updated as[1, 2]:

$$\omega^f \leftarrow \omega^f P(\mathbf{x}_{td}|\mathbf{z}^f_{td}, s^f_{td}, \mathbf{x}_{1:t-1}), \quad (7)$$

which has the intuitive explanation that the weight for particle f is updated by multiplying in the marginal probability of the new observation $\mathbf{x}_{td}$, 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.

Hyperparameters: The hyperparameters of the model are α, β_0, ϕ_0, and Ω_0. For our experiments we set $\beta_0 = .1$,

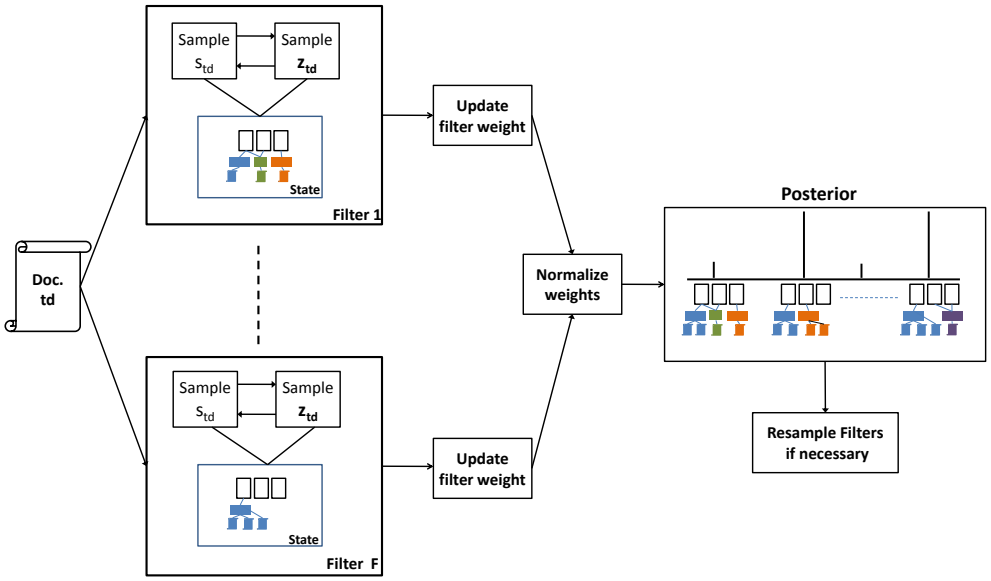

Figure 2: Illustration of the particle filtering algorithm. The state of each filter represents the hypothesis maintained by each particle before processing document td. The state is shown as a 3-layer hierarchy: top layer is the topics, then the stories and finally the documents. Each story maintains a distribution over topics, and each document is associated with a single story. Each filter runs an MCMC algorithm over each document in parallel and then update the filter weight. The filters' weights are then normalized to arrive at the posterior distribution shown to the right of the figure.

$\phi_0 = 0.01$, and $\Omega_0 = .001$. The Dirichlet prior α is uniform over topics but with a different value for the storyline topic. That is, we choose $\frac{0.1}{K+1}$ for the high-level topics and $\frac{0.5}{K+1}$ for the storyline specific term. For the RCRP, we allow the new-storyline hyperparameter γ_t to be epoch-specific, we apply a Gamma(5,10) prior and sample after every batch of 20 documents. See [17] for details. As for the kernel parameters, we set $\Delta = 3$ and $\lambda = 0.5$ — results were robust across a range of settings. For all experiments, we use 8-particles running on an 8-core machine.

4.2 Implementation and Storage

Implementing our SMC algorithm for large datasets poses runtime and memory challenges. To address these, we must first understand the algorithm's requirements. Operations in our algorithm can be divided into two categories: in the first category, we have particle filtering operations that work on single particles; these operations constitute the bulk of our algorithm. Because these particles can be worked on individually, we spawn one worker thread per particle to achieve fast parallel inference. The second category contains just the particle resampling operation. This operation is fundamentally different from the others, as it involves copying entire particles rather than simple updates to individual particles. We handle particle resampling using a master thread, during which all other threads sleep.

Given these needs, our algorithm requires a data structure that supports fast updates of individual particles' data, as well as fast copying of particles. In particular, the latter ensures that the time spent in the master thread is negligible compared to the worker threads, giving our implementation high parallel efficiency. It should be obvious that the naive implementation, where each particle has its own set of arrays for storage, is wholly unsuited for the latter operation — our runtime would be crippled by incessant memory copying. Worse, memory requirements would grow linearly in the number of particles, making large data streams impractical even for modest numbers of particles.

Inheritance trees: We overcome these problems with an idea from Canini *et al.* [12], 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 3 for an example. With tree maintenance, data retrieval becomes a practically constant time operation.

Thread safety: Thus far, we have only described Canini *et al.*'s version of the inheritance tree [12], which is *not* thread-safe. To see why, consider what happens when particle 1 is

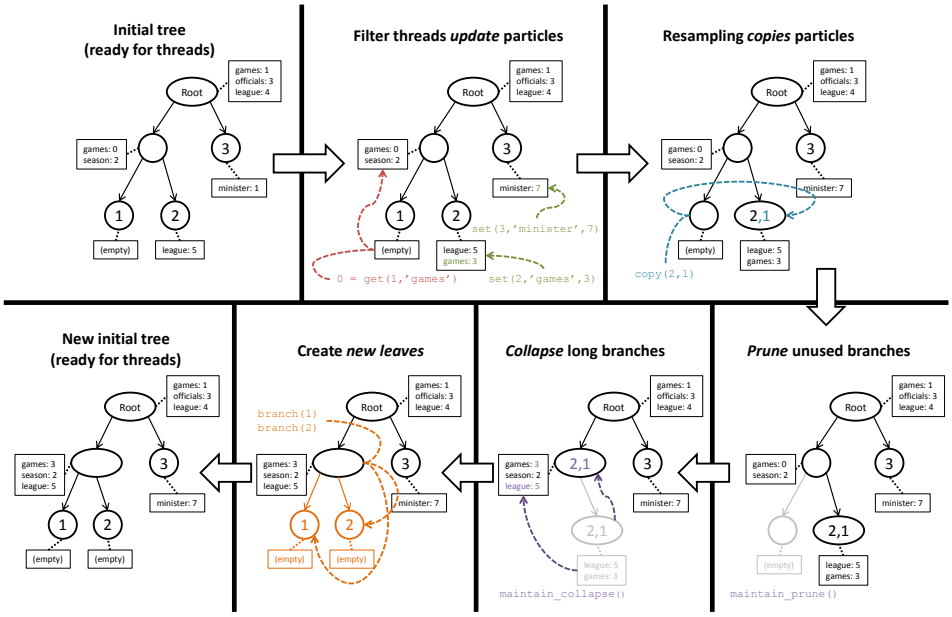

Figure 3: Inheritance tree operations in the context of our SMC algorithm. Numbers within a vertex represent associated particles. Each vertex's hash map is represented by a table, connected by a dotted line.

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 because our algorithm is multi-threaded.

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's vertex is an ancestor of any other particle's vertex, and writes only go to the assigned vertex, never to ancestors. Furthermore, every particle is associated with only one worker 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 we handle with a master thread (all other threads become inactive). So 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:

1. A `branch(f)` operation that creates a new leaf for a given particle f.
2. Update operations `get(f,i)` and `set(f,i,value)` that retrieve or write to data elements i for particle f.
3. A `copy(f,g)` operation that copies particle f to g.
4. A `maintain()` operation that prunes particle-less branches and then collapses unnecessary long branches.

These operations are demonstrated in Figure 3. The worker threads only use update operations, while master thread resampling uses the copy operation. After resampling but before restarting the worker threads, the master thread invokes the maintain operation to reduce the inheritance tree's size, followed by the branch function on every particle to make the tree thread-safe.

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 (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 support these operations we implemented an extended inheritance tree that integrates the our thread-safe inheritance tree with the inverted representation of [31]. Details of this novel data structure and its incorporation with the sampler can be found in [1, 2]

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 on-line execution. Moreover, our bounded memory requirements allow us to handle datasets of the scale seen in our experiments.

5. EXPERIMENTS

Our experiments assess our approach for both accuracy and scalability. Our goal is to provide the user with a representation of news stories that would allow them to find relevant stories easily and satisfactorily. We consider two datasets: news articles from Yahoo! and the TDT5 dataset[26]. We first describe the two datasets and then give three evaluations: clustering accuracy, structured browsing, and finally, an ablation study to understand the contribution of every component of our model.

5.1 Corpora

Yahoo! News Dataset.

We examine our model on English news samples of varying sizes extracted from Yahoo! News over a two-month

period. Details of the news samples are listed in Table 1. We use a sophisticated named entity recognizer [34] which disambiguates and resolves named entities to Wikipedia entries in the data preprocessing step. In addition, we remove common stop-words and tokens which are neither verbs, nor nouns, nor adjectives from the news articles. For the purpose of modeling, we divide each of the samples into a set of 12-hour epochs according to the article publication date and time.

TDT5 Dataset.

The Topic Detection and Tracking dataset (TDT) [26] contains several hundred thousand news articles, along with storyline relevance judgments from human raters. We use a month of data from the TDT5 corpus, comprising 46,793 documents. Of these, 1771 are annotated as belonging to one of 37 storylines; the others do not participate in evaluation, but must still be processed and should not distract the system from properly processing the annotated documents. Named entities are extracted using the Stanford NER system [18]. After applying a stoplist of 500 words [28], the vocabulary of both words and named entities is pruned to the 3000 most frequent terms.

5.2 Evaluating Clustering Accuracy

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

Table 1: Details of Yahoo! News samples and corresponding clustering accuracies of our method (Story) and the baseline (LSHC) evaluated 2525 labeled must-link and cannot-link pairs.

No.	Sample size	# Words	# Entities	Story Accuracy	LSHC Accuracy
1	111,732	19,218	12,475	**0.801**	0.738
2	274,969	29,604	21,797	**0.806**	0.791
3	547,057	40,576	32,637	**0.817**	0.800

For the sake of evaluating clustering, we compare against a variant of a single-link clustering baseline [14]. This simple baseline is the best performing system on TDT2004 task and was shown to be competitive with Bayesian models [33]. This method scales linearly with the number of all previously seen documents, however, in [25], 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 [20, 19], 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. In the remainder of this paper, we simply refer to this baseline as LSHC.

We ran both our method and the baseline on the datasets listed in Table 1. For our method, we used 200 topics.

From Table 1, we see that our method outperformed the baseline on all the datasets tested. We believe that this outcome stems from the fact that the baseline method does not make use of any statistical topic-document models that could help link two articles related to the same story but written with rather different wordings. Note that all clustering algorithms working at the surface-word level would suffer from this problem. Moreover, the fact that our *online, single-pass* algorithm is competitive with an *off-line* algorithm, is an encouraging result due to the streaming nature of our application.

Table 2: Clustering accuracies of our method on various samples vs. number (K) of topics.

No.	K=50	K=100	K=200	K=300
1	0.778	**0.809**	0.801	0.800
2	0.754	0.795	**0.806**	0.792

To study the importance of the number of topics to our model, we performed another set of experiments with different numbers of topics. Table 2 shows that the number of topics matters in the clustering accuracy of our method but it becomes less of a concern if we use a sufficient number of topics – see [2, 1] for more details.

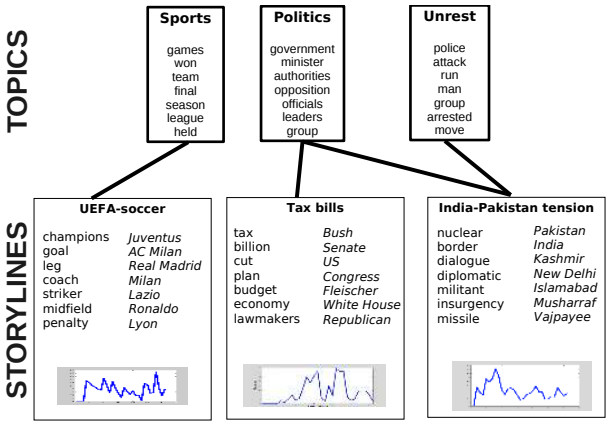

Figure 4: Some example storylines and topics extracted by our system. For each storyline we list the top words in the left column, and the top named entities at the right; the plot at the bottom shows the storyline strength over time. For topics we show the top words. The lines between storylines and topics indicate that at least 10% of terms in a storyline are generated from the linked topic.

5.3 Structured Browsing

We illustrate the utility of our model by describing some of the topics and stories and then showing how this representation can support structured browsing. The storylines in Figure 4 include the UEFA soccer championships, a tax bill under consideration in the United States, and tension between 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.

One way for end-users to take advantage of the organi-

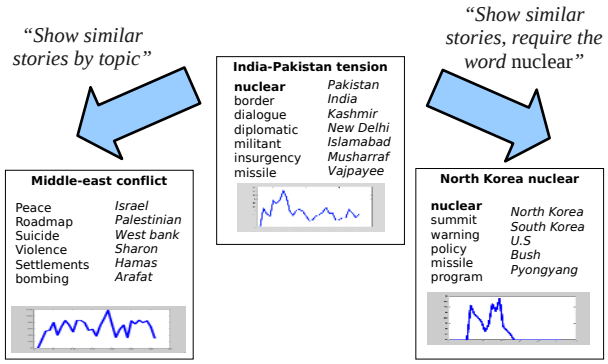

Figure 5: An example of structure browsing affordances provided by our model. Starting with the story about India-Pakistan tension, the user may request similar stories by topic, obtaining a story about the Middle East conflict (left), which shares topics such as politics, unrest and diplomacy. Alternatively the user may request topically similar stories, but require that the term "nuclear" be present — this leads to a storyline about the North Korean nuclear weapons program.

Table 3: Evaluation of complete model, with varying number of topics. Lower scores are better.

# Topics	1	30	50	100	200
C_{det}	0.80	0.79	0.76	**0.714**	0.75

Table 4: Ablation test, removing key features from the 100-topic model.

	ablated feature	C_{det}
1	time	0.74
2	entities	0.90
3	topics	0.81
4	storyline word distributions	0.75

zation obtained by our model is to browse the collection of high-level topics and then descend to specific stories indexed under each topic — like opening a physical newspaper and going directly to the sports page. However, our model provides a number of other affordances for structured browsing which were not possible under previous approaches. Figure 5 shows two examples. First, users can request similar stories by topic: starting from a story about India-Pakistan tension, the system returns a story about the Middle-east conflict, which also features topics such as diplomacy, politics, and unrest. In addition, users can focus their query with specific keywords or entities: the right side of Figure 5 shows the result of querying for similar stories but requiring the keyword *nuclear* to have high salience in the term probability vector of any story returned by the query. Similarly, users might require a specific entity — for example, accepting only stories that are a topical match and include the entity *Vajpayee*. 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.

5.4 Evaluating Individual Model Components

We evaluate against human-annotated storylines in the TDT5-May dataset to assess the contribution of specific components of our system: the recurrent Chinese restaurant process for modeling time, multilevel modeling of topics and storylines, and the incorporation of named entities. We assess the importance of each component by measuring the effect of removing it on the accuracy of the "first story detection" task, which is the problem of correctly identifying the first document in each annotated storyline; this is considered to be the most difficult problem in the TDT task [7]. We use the TDT scoring software and report the standard metric, which is the macro-averaged minimum c_{det} cost. Our system outputs the probability that each story is new, this probability is taken to be $P(s_{td} = new)$ averaged over particles. The minimum c_{det} cost is a measure that combines both false alarms and misses in a single number. **Lower** numbers are better [26]. Evaluation is performed in a *streaming* setting, meaning that the system must make a judgement for each document shortly after it is received – rather than after viewing the entire dataset.

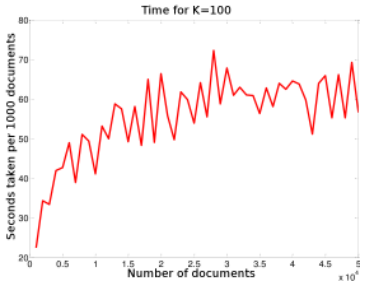

Figure 6: Time Scalability

Our quantitative evaluation of storyline quality is shown in Tables 3 and 4. Experimenting with a range of topics, we find the strongest results around 100. The average time to process one document ranged from **50ms** for $K = 50$ to **80ms** for $K = 200$ thanks to our effective extended inheritance tree. Fixing $K = 100$, we then tested the contributions of each feature of our model (Table 4).

Each key aspect of our system improves performance: the two-level topic/storyline model strongly outperforms a model that ignore topics (line 3 of Table 4); the `RCRP` model outperforms the `CRP` model, demonstrating the importance of modeling time (line 1); named entities improve performance (line 2) and the story specific language model also enhances outcome (line 4).

Our second evaluation concerns scalability. Our goal is a system that can be deployed over a long time period, continually processing data with time and memory costs that are constant in the length of the history. As shown in Figure 6, our system scales well as the history grows: after an initial build-up period, the time taken by our system to process 1000 documents increases very slowly as the total number of documents processed by the system increases. After the build-up period, the average time to process a document stabilized around 60 ms per document for $K = 100$ (the residual growth is due to the increasing number of stories). Finally we would like to mention that our method is completely unsupervised, in contrast to many TDT systems which tune their parameters over a training dataset from an earlier TDT run. Our use of TDT5 here was merely to evaluate the contribution of each component of our model.

6. RELATED WORK

Modeling: Our approach is non-parametric over stories, allowing the number of stories to be determined by the data. Similarly, Zhang et al. [33] describe an online

clustering approach using the Dirichlet Process, which they evaluate on the same first-story detection task that we consider here. This work equates storylines with clusters, and does not model high-level topics. Non-parametric clustering has been combined with topic models, with the cluster defining a distribution over topics [32, 29]. 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.

There have been several efforts to augment topic models with time [10, 30]. These approaches allow topic strength to vary smoothly; however, they do not incorporate a multi-level model that distinguishes individual storylines from high-level topics, and thus must treat topic evolution as a relatively slow phenomenon. Consequently, such models have been applied successfully to long-term corpora (over several years or even decades), but have not been shown to successfully model rapidly-changing corpora such as news or blogs.

Inference: Recent work on topic models has focused on improving scalability; we focus on sampling-based methods, which are most relevant to our approach. Yao et al. use Gibbs sampling but do not resample the topics for documents outside a fixed temporal window [31]; we differ by developing a Monte Carlo sampling approach that handles the streaming setting explicitly. Banerjee and Basu also develop online sampling equations, but they instantiate the parameters θ and ϕ whereas we marginalize them [9]. Our approach is most influenced by the particle filter of [12], from which we take the idea of efficiently storing the hidden variable history.

7. CONCLUSION

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 and non-parametric model. We see many promising directions for future research. One particularly appealing possibility would be a hierarchical nonparametric model of topics and storylines that features multiple levels of increasing detail.

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.

8. REFERENCES

[1] —-. The online infinite topic-cluster model: Storylines from streaming text. *AISTATS*, 2011. Under review.

[2] A. Ahmed,Q. Ho, C. Teo, J. Eisenstein, A. J. Smola, E.P. Xing The online infinite topic-cluster model: storylines from streaming text. *CMU-ML-11-100*, 2011.

[3] A. Ahmed and E.P. Xing. Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *SDM*, 2008.

[4] N. Ailon, M. Charikar, A. Newman Aggregating inconsistent information: Ranking and clustering In *Journal of ACM*, 55(5):1–27, 2008.

[5] J. Allan. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer, 2002.

[6] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *DARPA News Understanding Workshop*, 1998.

[7] J. Allan, V. Lavrenko, and H. Jin. First story detection in TDT is hard. In *CIKM*, 374–381, 2000.

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

[9] A. Banerjee and S. Basu. Topic models over text streams: A study of batch and online unsupervised learning. In *Proceedings of SDM*, 2007.

[10] D. Blei and J. Lafferty. Dynamic topic models. In W. W. Cohen and A. Moore, editors, *ICML*, 2006.

[11] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[12] K. R. Canini, L. Shi, and T. L. Griffiths. Online inference of topics with latent dirichlet allocation. In *AISTATS*, 2009.

[13] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, 2009.

[14] M. Connell, A. Feng, G. Kumaran, H. Raghavan, C. Shah, and J. Allan. UMass at TDT 2004. In *TDT 2004 Workshop Proceedings*, 2004.

[15] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[16] G. Doyle and C. Elkan. Accounting for burstiness in topic models. In *ICML*, 2009.

[17] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *JASA* 90, 1995.

[18] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370, 2005.

[19] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *VLDB*, 1999.

[20] T. Haveliwala, A. Gionis, and P. Indyk. Scalable Techniques for Clustering the Web. In *WebDB*, 2000.

[21] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *SIGIR*,2004.

[22] D. M. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *UAI*, 2008.

[23] T. P. Minka. Estimating a Dirichlet distribution. Technical report, MIT, 2003.

[24] D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *KDD*, pages 680–686, New York, NY, USA, 2006.

[25] S. Petrovic, M. Osborne and V. Lavrenko. Streaming First Story Detection with application to Twitter. In *NAACL*, 2010.

[26] NIST. `http://www.itl.nist.gov/iad/mig/tests/tdt/2004/workshop.html`.

[27] J. Pitman. Exchangeable and partially exchangeable random partitions. *Probability Theory*, 102(2), 1995.

[28] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[29] H. Wallach. Structured topic models for language. PhD Thesis. Cambridge, 2008.

[30] X. Wang and A. McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *KDD*, 2006.

[31] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, pages 937–946, 2009.

[32] K. Yu, S. Yu, and V. Tresp. Dirichlet enhanced latent semantic analysis. In *AISTATS*, 2005.

[33] J. Zhang, Y. Yang, and Z. Ghahramani. A probabilistic model for online document clustering with application to novelty detection. In *NIPS*, 2004.

[34] Y. Zhou, L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney. Resolving Surface Forms to Wikipedia Topics. In *COLING*, 1335–1343, 2010.