Uncovering Hidden Structure in Sequence Data via Threading Recurrent Models

Manzil Zaheer, Amr Ahmed, Yuan Wang, Daniel Silva, Marc Najork
Google AI
Mountain View, CA
(manzilz,amra,yuanwang,dansilva,najork)@google.com

Yuchen Wu, Shibani Sanan, Surojit Chatterjee
Google
Mountain View, CA
(yuchenwu,ssanan,schatterjee)@google.com

ABSTRACT

Long Short-Term Memory (LSTM) is one of the most powerful sequence models for user browsing history [17, 22] or natural language text [19]. Despite the strong performance, it has not gained popularity for user-facing applications, mainly owing to a large number of parameters and lack of interpretability. Recently Zaheer et al. [25] introduced latent LSTM Allocation (LLA) to address these problems by incorporating topic models with LSTM, where the topic model maps observed words in each sequence to topics that evolve using an LSTM model. In our experiments, we found the resulting model, although powerful and interpretable, to show shortcomings when applied to sequence data that exhibit multi-modes of behaviors with abrupt dynamic changes. To address this problem we introduce thLLA: a threading LLA model. thLLA has the ability to break each sequence into a set of segments and then model the dynamic in each segment using an LSTM mixture. In that way, thLLA can model abrupt changes in sequence dynamics and provides a better fit for sequence data while still being interpretable and requiring fewer parameters. In addition, thLLA uncovers hidden themes in the data via its dynamic mixture components. However, such generalization and interpretability come at a cost of complex dependence structure, for which inference would be extremely non-trivial. To remedy this, we present an efficient sampler based on particle MCMC method for inference that can draw from the joint posterior directly. Experimental results confirm the superiority of thLLA and the stability of the new inference algorithm on a variety of domains.

ACM Reference Format:
Manzil Zaheer, Amr Ahmed, Yuan Wang, Daniel Silva, Marc Najork, Yuchen Wu, Shibani Sanan, and Surojit Chatterjee. 2019. Uncovering Hidden Structure in Sequence Data via Threading Recurrent Models. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3291036

1 INTRODUCTION

Modeling sequential data with discrete observations is an important problem in machine learning with applications ranging from biology [10] to text [19] to user behavior [16]. For instance, in case of natural language text, the aim commonly is to predict the next word given the context. Likewise, in case of user activity modeling the aim would be to predict the next action of the user given the history in an interpretable fashion so as to serve and target relevant and useful contents to users. It has been found that a good sequential data model should be accurate, sparse, and interpretable [25]. Unfortunately, few models that satisfy these requirements exist.

Recurrent neural networks (RNN) [18], such as LSTMs (Long-Short Term Memory) [13] have become the state-of-the-art in modeling sequential data. While effective and expressive, [15] they result in a generally uninterpretable and inaccessible representation to humans [21]. Moreover, the number of parameters of the model is proportional to the size of observation space which further diminishes accessibility to humans. For use-cases where interpretability is of utmost importance, latent variable models such as LDA [5] have been deployed with good commercial success [14]. Although such topic models are not a sequence model, they are powerful tools for uncovering latent structure [1].

Recently, [25] proposed Latent LSTM Allocation (LLA) that bridges the gap between LSTM and LDA. In LLA words are grouped into topics using LDA-like technique, then the dynamic evolution of topics in the sequence is captured using an LSTM-like technique. The model is fit jointly by alternating a sampling step to fit the topic and an optimization step to update the parameters of the LSTM. The resulting model enjoys the strengths of both LDA and LSTM: it is sparse, interpretable and requires fewer parameters that scales with the number of topics rather than size of observation space.

Notwithstanding these developments, LLA, while expressive and interpretable, can still capture spurious dynamics in sequential data with mixed themes. In Fig. 1 we visualize the learned transition graph over topics as extracted from LLA when fitted over a user search history dataset. As evident, while the model captured useful transitions between camping gear and hiking topics, it also captured a couple of spurious transitions between ice-cream and painting topics. This can be attributed to mixed nature of user data, for instance, the user might search for hiking, home remodeling, and baking desserts with abrupt transitions between these totally different themes. While LSTM has the expressive power to capture long-range dependency structures, the introduction of latent variables (topics) exacerbate the complexity of the surface of objective function and precludes the optimization algorithm from linking each item in the sequence to the most relevant item in the the same theme from the user history. In addition LLA can not break the transition graph into themes such as camping and baking deserts that would give another layer of structure and interpretability.

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02.
https://doi.org/10.1145/3289600.3291036

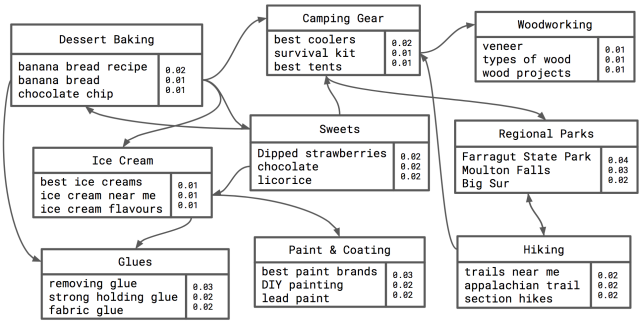


Figure 1: Transition graph from LLA illustrating its shortcoming. The LLA model is fit over user search history data, where each topic is a distribution of search queries and each edge denotes a temporal transition. As evident, LLA captures both useful and spurious dynamics.

To address these shortcomings of LLA, we introduce thLLA, threading LLA as a more faithful model for sequential data with mixed themes and abrupt transitions between the themes. thLLA generates a sequence from a mixture of LLA models, topics are shared among all LLA models in the mixture, however, each LLA model in the mixture has its own LSTM component that captures dynamic evolution over a subset of the topics in the shared space. The input data is endowed with a switch variable for each item that selects a given mixture to generate this item. In that regard each item in the sequence is generated NOT conditioned on all the previous items as in LLA but rather conditioned on items that are associated with the same mixture in the history. In that regard our model can be seen as a threading or segmentation model that breaks the user sequence into coherent themes and models dynamic evolution within each theme separately. thLLA still enjoys all nice properties of LLA, namely, interpret-ability and small model size while being more accurate. Unfortunately, this comes with an added complex dependency structure for which inference would be extremely non-trivial. To remedy this, we present an efficient sampler based on particle MCMC method for inference that can draw from the joint posterior over both segments and topic assignments. We show quantitatively as well as qualitatively the efficacy and *interpretability* of thLLA over several datasets.

2 BACKGROUND

The proposed thLLA model and its inference builds on the concepts of recurrent neural networks, LLA and sequential Monte Carlo. In this section, we provide a brief review of these building blocks.

RNNs is a powerful tool for modeling sequential data due to their strong performance and ease of deployment. Technically, a RNN is a automaton described by the triplet (Σ, S, δ):

- $\Sigma \subseteq \mathbb{R}^D$ is the input alphabet
- $S \subseteq \mathbb{R}^H$ is the set of states
- $\delta : S \times \Sigma \rightarrow S$ is the state transition function made up of a neural network.

Given a sequence $x_1, x_2, ..., x_T \in \Sigma^*$, a run of the RNN consists of reading the input x_t at a time step and updating its state $\mathbf{s}_t$ by applying the transition function δ on the previous state $\mathbf{s}_{t-1}$ and the input, i.e. $s_t = \delta(s_{t-1}, x_t)$. Owing to the flexibility transition function because of neural networks, RNNs can express the complex dynamics present in sequence data.

In case of discrete sequence modeling, the input and output is not in a format usable by RNN. The discrete input symbol is handled the by mapping to vector in Σ by using a lookup table E. This can be costly when size of input symbols, i.e. $|\Sigma|$ is large. The desired output, which is the probability of x_{t+1}, e.g. next user action, is produced by transforming the state $\mathbf{s}_t$:

$$p(x_{t+1}|x_{1:t}) = g(\mathbf{s}_t), \tag{1}$$

where g is an appropriate differentiable function. Like any other neural neural network, RNNs can be trained using back propagation.Although RNN can, in principle, model long-range dependencies it can suffer from the problem of exploding or vanishing gradient. To mitigate this to some degree, LSTMs [13] were introduced as a special case of RNN.

In practice, LSTMs were shown to be effective at capturing long and short patterns in data [15], however, it does not produce interpretable models and it requires a large number of parameters that scales with the size of the input vocabulary. To remedy this problem, [25] introduced LLA: Latent LSTM allocation, that combines ideas from both LSTM and topic models such as LDA. In LLA, each input sequence is mapped to a sequence of topics using an LDA-like model and an LSTM model learns the dynamic evolution over the topic space (rather than the input vocabulary space). This results in an interpretable model due to the use of LDA topics that group the input alphabet into sparse topics. Furthermore, LLA requires fewer parameters that scales with the number of topics not the size of the input alphabet.

Finally, we need a tool to sample from complex posteriors which would arise unavoidably in models having combination of LSTMs and Bayesian components. Sequential Monte Carlo (SMC) [9] is an algorithm that samples from a series of potentially unnormalized densities $\nu_1(z_1), \ldots, \nu_T(z_{1:T})$. At each step t, SMC approximates the target density ν_t with P weighted particles using importance distribution $f(z_t|z_{1:t-1})$:

$$\nu_t(z_{1:t}) \approx \hat{\nu}_t(z_{1:t}) = \sum_p \alpha_t^p \delta_{z_{1:t}^p}(z_{1:t}), \tag{2}$$

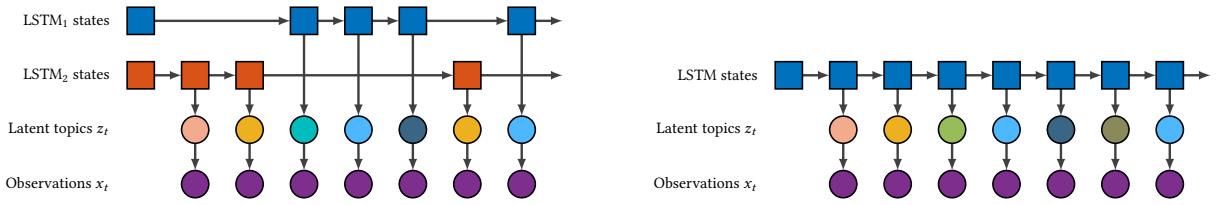
where α_t^p is the importance weight of the p-th particle and δ_x is the Dirac point mass at x. Repeating this approximation for every t leads to the SMC method, outlined in [3, 9].

3 MODEL

In this section, we provide a detailed description of the proposed thLLA model. thLLA inherits the good properties from LLA of sparsity and interpretability allowing human analysis of the components but adds the ability to accurately track events especially in the presence of multiple modes of behavior.

3.1 Generative process

In thLLA, we employ a mixture of different LSTM components to model different segments of the dynamics in each sequence. Switching between the mixture components is modeled in the topic space like LLA and we use a multinomial-Dirichlet distribution to model emissions $p(x_t|z_t)$ similar to LDA, where x_t is the observed sequence and z_t is the latent topic index.



(a) Example of thLLA: Can easily switch context

(b) Corresponding LLA struggles during abrupt switch

Figure 2: Illustration of generative process of thLLA vs LLA. In this example, thLLA breaks the sequence into two segments: one about blue topics (camping) and one about red topics (food and desert). It models each segment with separate LSTM that can capture clean, intricate dynamics over these subset of topics, whereas LLA struggles in practice. Note that, in LLA, topic at time t is linked to that at time $t-1$ but in thLLA, each time step is linked to the most recent time step in its theme (not necessarily the most recent one in time).

Suppose we have K topics and M mixtures in the dynamics. For each of the M mixtures, we have different LSTMs, denote them by $\text{LSTM}_1, ..., \text{LSTM}_M$. All the LSTMs share a common embedding look-up table E. Assume that We have a dataset of user activity histories $\mathcal{D}$ where each user history $u \in \mathcal{D}$ is composed of N_u actions. With these notations in place, the complete generative process can be described as follows:

(1) **for** $k = 1$ **to** K:
  (a) Draw topic distribution $\boldsymbol{\phi}_k \sim \text{Dir}(\beta)$
(2) **for each** user u **in** dataset $\mathcal{D}$:
  (a) Initialize all LSTMs with $\mathbf{s}_m = 0$ and set $z_m^{\text{prev}} = 0$ for $0 \le m \le M$
  (b) Draw user $\boldsymbol{\pi}_u \sim \text{Dir}(\alpha)$
  (c) **for each** action index $t = 1$ **to** N_u:
    (i) Select behavior mode at time t as $y_t \sim \text{Categorical}(\boldsymbol{\pi}_u)$
    (ii) Update corresponding dynamics $\mathbf{s}_{y_t} = \text{LSTM}_{y_t}\left(E\left[z_{y_t}^{\text{prev}}\right], \mathbf{s}_{y_t}\right)$
    (iii) Get topic proportions at time t from the active LSTM state, $\boldsymbol{\theta} = \text{softmax}_K(\mathbf{W}_p \mathbf{s}_{y_t} + \mathbf{b}_p)$
    (iv) Choose a topic $z_{u,t} \sim \text{Categorical}(\boldsymbol{\theta})$
    (v) Save $z_{y_t}^{\text{prev}} = z_{u,t}$
    (vi) Choose user action $x_{u,t} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{u,t}})$

For a single user history sequence u, the above mentioned generative process specifies the following marginal probability distribution of observing the sequence:

$$p(x_u|\omega, \boldsymbol{\phi}, \boldsymbol{\pi}_u) = \sum_{y_u, z_u} p(x_u, y_u, z_u|\omega, \boldsymbol{\phi}_u, \boldsymbol{\pi}_u)$$
$$= \sum_{y_u, z_u} \prod_t p(z_{u,t}|z_{u,0:t-1}, y_{u,1:t}, \omega) p(y_{u,t}|\boldsymbol{\pi}_u) p(x_{u,t}|\boldsymbol{\phi}_{z_{u,t}}) \tag{3}$$

Here ω is the union of parameters of all $\text{LSTM}_1, ..., \text{LSTM}_M$, the embedding matrix E, the projection matrix W_p and bias b_p. The structure of the generative process/probability function and importance of the switch variable y_t is better illustrated in Figure 2 through an example. Here each element of the observed sequence x_t is associated with a topic z_t and a switch variable y_t where y_t links (x_t, z_t) to the elements in the history with the same mode of behavior (theme). For instance in Figure 2, we have two themes in this sequence, one is given by topics with yellow shades (e.g camping) and the other is defined by topics with blue shades (e.g baking deserts). The dynamic evolution of each theme is modeled using a different LSTM mixture. In that regard, the switch variable introduces large jumps in the sequence to connect relevant items so that data point (x_t, z_t) is generated conditioned only on elements from the same theme in the history $(x_{1,t-1}, z_{1,t-1})$, whereas LLA generates this element conditioned on all elements in the history. While in theory LSTM should be able to figure out these jumps, due to the diminishing gradient problem, and the fact that these jumps can occur weeks apart, the optimization algorithm in practice misses them as depicted in Figure 2 via the missing yellow topic in the second-to-last position.

Note that LLA [25] is a special case of thLLA when the number of mixtures $M = 1$. thLLA still enjoys all good properties of LLA, namely sparsity, fewer parameters, and interpretability. In addition, the introduction of different dynamic mixtures adds another layer of interpretablility by breaking the data into a set of themes.

3.2 Parameter estimation

Exact inference for thLLA is intractable due to complex dependences structure. As a result we resort to an approximate technique like mean field variation inference [24] or stochastic EM [26]. We begin by writing down, the variational lower bound to the marginal data likelihood is given by

$$\sum_u \log p(\boldsymbol{x}_u|\omega, \boldsymbol{\phi}, \boldsymbol{\pi}_u)$$
$$\ge \sum_u \mathbb{E}_q \left[\log \frac{p(z_u|y_u, \omega) p(y_u|\boldsymbol{\pi}_u) p(x_u|z_u, \boldsymbol{\phi})}{q(\boldsymbol{y}_u, z_u)}\right], \tag{4}$$

where q is the variational distribution. Following the (stochastic) EM approach, iteratively maximizing the lower bound w.r.t. q and the model parameters $(\omega, \pi, \boldsymbol{\phi})$ leads to the following updates:

- **E-step**: The optimal variational distribution for each user u is given by the posterior:

$$q^\star(\boldsymbol{y}_u, z_u) \propto p(z_u|\boldsymbol{y}_u, \omega) \prod_t p(y_{u,t}|\boldsymbol{\pi}_u) p(x_t|\boldsymbol{\phi}_{z_{u,t}}). \tag{5}$$

  Unlike Markov models for which efficient smoothing algorithms such as the forward-backward algorithm are available for computing the posterior expectations of sufficient statistics, in case of thLLA, although the forward messages can still be computed,

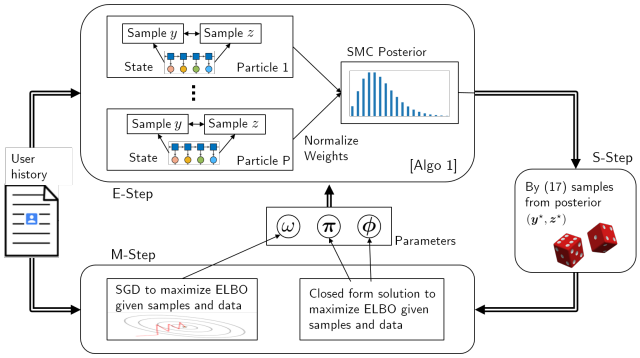


Figure 3: Overview of the Stochastic EM inference algorithm. Details are given in the Appendix.

the backward recursion can no longer evaluated or efficiently approximated.

- **S-step**: As taking expectations is difficult, we take an alternative approach to collect posterior samples for every user u instead:

$$(\boldsymbol{y}_u^\star, \boldsymbol{z}_u^\star) \sim q^\star(\boldsymbol{y}_u, \boldsymbol{z}_u), \tag{6}$$

given only the filtering equations and SMC. We discuss the posterior sampling algorithm in detail in the appendix.

- **M-step**: Given the posterior samples $(y_u^\star, z_u^\star)$ for all users u, which can be seen as Monte Carlo estimate of the expectations, the subproblem for ω, $\boldsymbol{\pi}$, and $\boldsymbol{\phi}$ are

$$\begin{aligned}
\omega^\star &= \operatorname*{argmax}_{\omega} \sum_u \log p(z_u^\star | y_u^\star, \omega), \\
\boldsymbol{\pi}_u^\star &= \operatorname*{argmax}_{\boldsymbol{\pi}_u} \sum_t \log p(y_{u,t} | \boldsymbol{\pi}_u) \quad \forall u \\
\boldsymbol{\phi}^\star &= \operatorname*{argmax}_{\boldsymbol{\phi}} \sum_u \sum_t \log p(x_{u,t} | \boldsymbol{\phi}_{z_{u,t}^\star}).
\end{aligned} \tag{7}$$

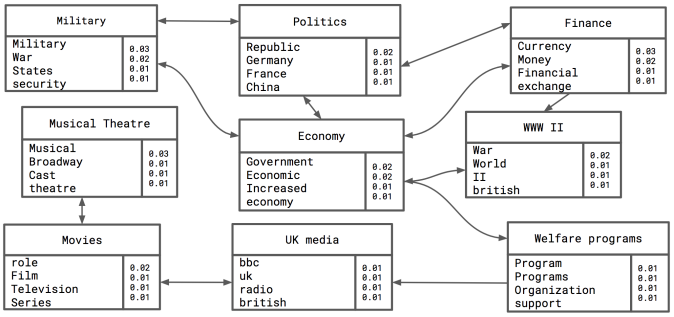
The first equation basically implies training each LSTM with sequences $z^\star$ corresponding to the time when its active as specified by $y^\star$. The other two equations are simply the MLE of switch variable, and the MLE of the given emission model, which is same as LDA or LLA. The full algorithm is summarized in Figure 3

4 EXPERIMENTS

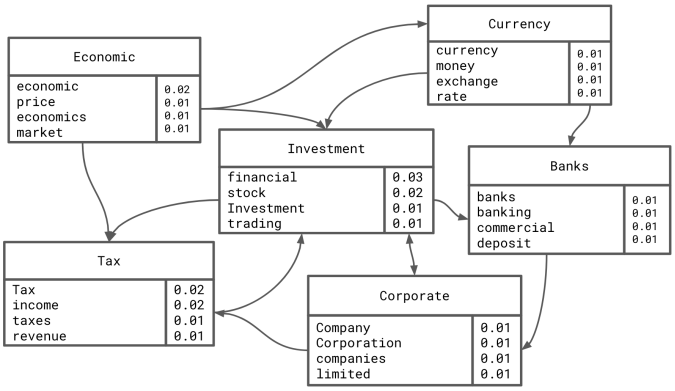
In this section we study empirically the properties of thLLA to establish that (1) it is flexible in capturing underlying nonlinear dynamics in sequences with mixed themes and abrupt jumps, (2) the inference algorithm is efficient (3) it produces cleaner and more interpretable topic transitions compared to LLA especially with the add layer of structure due to the dynamic mixture components.

4.1 Setup and Datasets

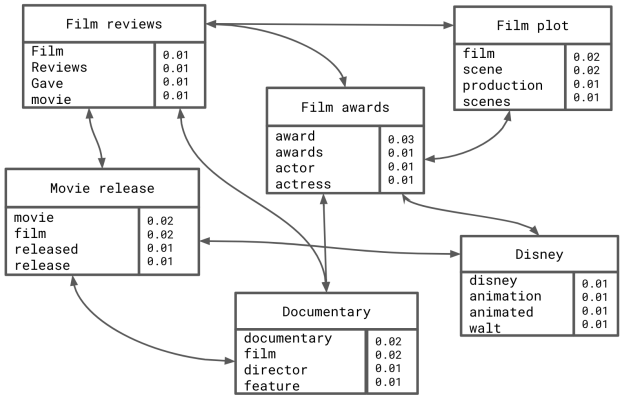
For all experiments we follow the standard setup for evaluating temporal models, i.e. we use the first 60% portion of the each sequence for training and then predict the remaining 40% of the sequence based on the representation inferred from the first 60%. We use perplexity as our metric (lower values are better) and compare against the baseline LLA. Since [25] established the superiority of LLA against other temporal variants of topic models such as [2, 4],



(a) Base LLA transition graph



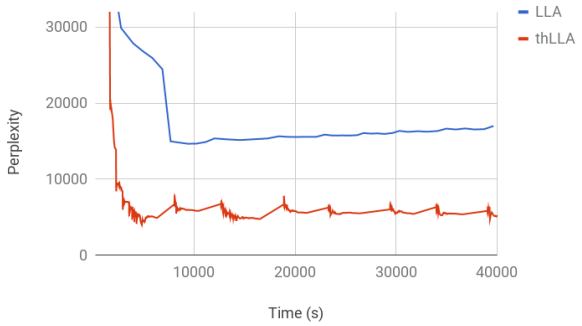
(b) thLLA transition graph for finance-related topics



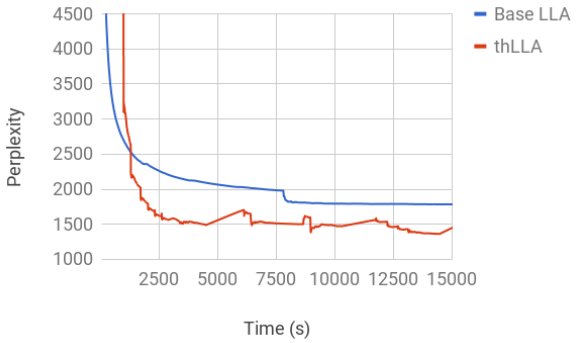
(c) thLLA transition graph for film-related topics

Figure 4: Transition graphs on Wikipedia dataset. Top row shows base LLA and bottom row depicts transition from two difference thLLA mixtures about Movies and Finance. As evident thLLA gives cleaner representation. Probabilities are trimmed to the first two decimal digits.

we **exclude** those comparison for space limitations Similarly we exclude comparisons against base LSTM as the same trade-off holds as in LLA [25]. We focus here on the effect of the added layer of structure. Unless otherwise stated we run our experiments with number of topics $K = 1000$ and number of mixtures $M = 20$ (for thLLA). For both, we chose the dimensions of the input embedding

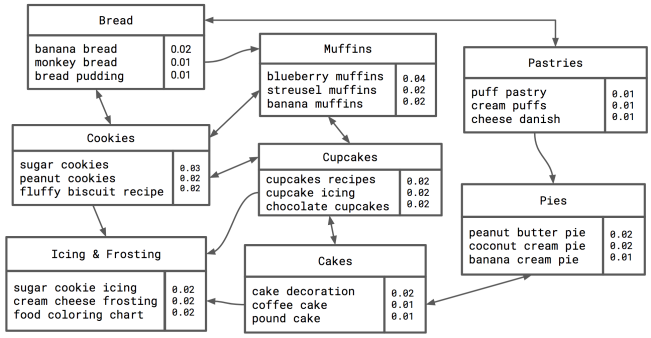


(a) User Search History

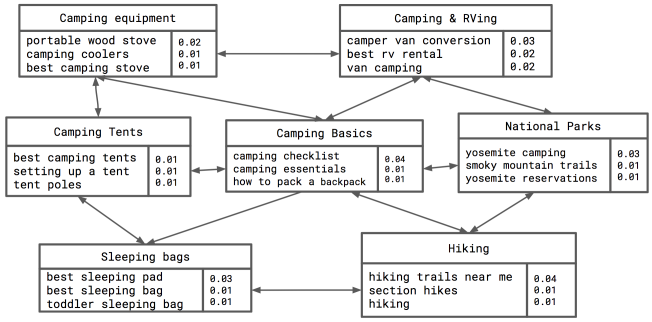


(b) Wikipedia

Figure 5: Converge curves for LLA and thLLA on a) User history and b) Wikipedia (lower better)



(a) Transition graph for dessert-related topics



(b) Transition graph for outdoors-related topics

Figure 6: Transition graphs on user history dataset. Figure depicts transition from two copa difference thLLA mixtures about Camping and Desert. As evident thLLA gives cleaner representation compared results obtained from LLA show in Figure 1. Probabilities are trimmed to the first two decimal.

to LSTM in the range {50, 100, 150} and the size of the evolving hidden LSTM state in the range of {50, ..., 500}. For thLLA, we share the embeddings of topics among all LSTM in the mixtures, leaving each mixture to only learn the evolving hidden LSTM state. For SMC inference, we gradually increase the number of particles P from 1 to K during training, where K is the number of topics. Finally, all the algorithms are implemented on TensorFlow and run on machines with 4 GPUs.

To establish our claims we use data of different characteristics:

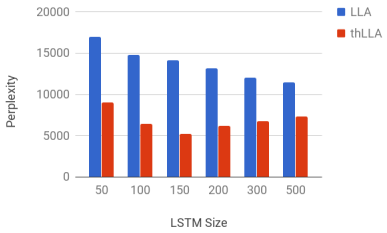
- **Wikipedia**: The publicly available Wikipedia dataset comprised of short documents containing 0.7 billion tokens and 1 million documents (Each document is a sequence). We chose the most frequent 50k words as a vocabulary. This domain is structured with each sequence having a few themes with no sharp transitions.
- **User Search History**: An anonymized sample of user search query history from a web search engine. For privacy reasons, we remove all queries appearing less than a given threshold, and use only head queries. This results in a dataset of 10M users and 1M vocabulary. This domain is less structured than the Wikipedia dataset with longer sequences (history of a given user), and abrupt jumps back and forth between themes in each sequence.

4.2 Qualitative Evaluation: Topic Graph

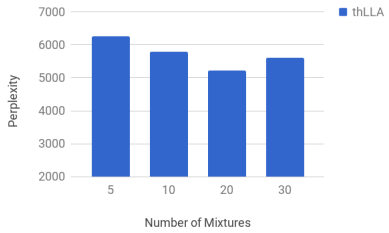
First we test the claim that thLLA gives cleaner and more interpretable representation compared to LLA. In Figure 4 and 6 we show the topic transition graph from both LLA and thLLA, each fit with 1000 topics and a comparable size of the LSTM(s) hidden state. For user history dataset the LLA transition graph is shown in Figure 1. There are various ways to illustrate the dynamic behavior learned by the LSTM component(s) in each model. For instance in [25], the authors applied hierarchical clustering over the input topic embedding from the LSTM component and showed that topics that appears closer in time also appears closer in the hierarchy. Here we make this more explicit and visualize the dynamics learnt by each model as a topic transition graph using the one step transition from the LSTM components, i.e for LLA we use $P_{LLA}(topic_t = k|topic_{t-1} = k'$, LSTM parameters) and for mixture m from thLLA, we use $P_{thLLA}(topic_t = k|topic_{t-1} = k'$, LSTM parameters of mixture m). In the above figures, we show top words (queries) for each topic as well as top transitions. As evident, while LLA still captures useful transition, the added layer of structure (themes) in thLLA results in a much cleaner models that uncovers fine-grained transitions such as between camping basics, equipments and parks, as can be seen in Figure 6(b), due to having a separate dynamic model for each mixture.

4.3 Quantitative Evaluation: Perplexity

Figure 5 compares LLA and thLLA quantitatively using held-out perplexity on both datasets. We use number of topics = 1000, for

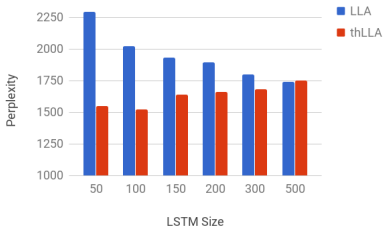


(a) Effect of LSTM hidden state size on User Data

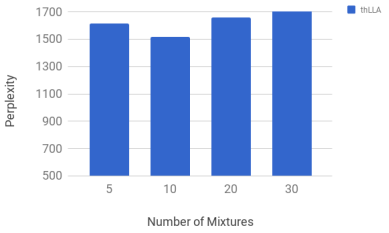


(b) Effect of Number of thLLA mixtures on User Data

(c) Effect of Number of topics on User Data



(d) Effect of LSTM hidden state size on Wikipedia Data



(e) Effect of Number of thLLA mixtures on Wikipedia

(f) Effect of Number of topics on Wikipedia Data

Figure 7: Ablation study comparing. From Left to right: effect of LSTM size, number of mixtures in thLLA, and number of topics. Top shows results on User Search History dataset and bottom shows results on the Wikipedia dataset.

thLLA number of mixtures = 10 and we fix the size of the hidden state of each LSTM mixture to be 50 while for LLA we use a value of 500. We use the same size for topic embedding in all models (100) and we share topic embedding across all mixtures in thLLA making the overall size of the two models comparable. First, overall thLLA give better results in both domain, however, the improvement is more pronounced over the user history dataset as expected due to its mixed-theme nature and abrupt transitions. Second, thLLA while being more complex still converges at the same rate compared to LLA. To understand this, we refer to [25] who noted that the time complexity of LLA training is dominated by the M-Step due to the LSTM component. Since thLLA utilizes smaller-sized LSTMs in its mixtures, this compensates for the extra complexity introduced in the E-step to sample both the topic and mixture assignment for each item in the sequence.

4.4 Effect of LSTM Size

In this section we preform an ablation study to show the effect of the LSTM size on the thLLA and LLA. One might argue that if we increase the size of the evolving LSTM hidden state in LLA we can capture the same effect introduced by thLLA, i.e the LSTM will figure it out due to its ability to model long-range dependencies. To test this hypothesis, we fix the number of topics to be 1000, the number of mixtures in thLLA to be 10, and LSTM topic embedding to be 100. We vary the size of the evolving hidden LSTM state in both LLA and each mixture in thLLA from 50 to 500. As shown in figure 7(a,d), this is not the case, and in fact thLLA outperforms LLA across the entire range especially in the User History dataset. In the Wikipedia dataset, thLLA peaks at around hidden sizes = 50-100 while LLA peaks around 500, given that we use 10 mixtures for thLLA we see that at comparable LSTM sizes thLLA still outperform LLA. One should note here that while the LSTM in LLA can theoretically model very long-range dependencies, in practice this is hard due to the diminishing gradient problem, thus especially for user sequences, the fact that users go back and forth between themes over weeks poses a real challenge to LLA. In contrast, in thLLA, each mixture considers only a small portion of the input sequence that share the same theme modeled by this mixture, thus two items might be adjacent in the input to the LSTM's mixture (thanks to the inference in the E-Step) while in fact they are separated by many steps in the observed sequence. This ability of thLLA to model long-range dependencies via jumps as explained in Section 3 is a key to it performance.

4.5 Effect of Number of Topics

In Figure 7(c,f) we show how varying the number of topics affect the quality of both models. We fix number of mixtures in thLLA to be 10 and use the best performing setting for LSTM configuration in each model. As shown in the figure thLLA outperform LLA across the entire topic range. Note that for the Wikipedia following [25] we vary number of topics in the range {250, 500, 1000} which does not change the overall trend.

4.6 Effect of Number of Mixtures

In Figure 7(b,e) we show how varying the number of mixtures in thLLA affects the model quality while fixing the number of topics to be 1000. As expected from the figure, this is a model selection problem where the performance peaks at 20 mixtures for the User Search history dataset. However, it should be noted that overall thLLA is not very sensitive to slight variation around the optimal value. In practice, cross validation over a development dataset can be used to find the best value of the number of mixtures.

Table 1: Query prediction on the user search history dataset. All improvements are statistically significant.

Model	P@5	P@10	P@15
LLA	0.174	0.151	0.139
thLLA	**0.219**	**0.192**	**0.183**

4.7 A User Prediction Task

In this section we provide an additional quantitative evaluation besides the standard preplexity metric using the user search history dataset. The task here is users' future query prediction. We divide queries in each test user into 80% for topic inference and 20% for evaluation. Each model produces a ranked list of queries based on the first 80% portion of the user history. We use precision@K to evaluate the accuracy of each model in predicting future queries where $P(\text{Query} = q|\text{user history})$ is computed as follows:

$$P^{\text{LLA}}(q|u) = \sum_{k=1}^{K} P^{LLA}(\text{topic} = k|u)P(q|k)$$
$$P^{\text{thLLA}}(q|u) = \sum_{m=1}^{M} \pi_{u,m} \sum_{k=1}^{K} P^{\text{thLLA},m}(\text{topic} = k|u)P(q|k) \quad (8)$$

In other words, in LLA we first run inference and use the last state of the LSTM to produce a future topic distribution that we combine with the topic-query distribution to produce the predicted ranked list of queries. In thLLA we perform the same computation on a per-mixture basis where the contribution of each mixture is weighted by its prevalence in the user history given by $\pi_{u,m}$. The results are shown in Table 1

As the table shows thLLA produced better results than LLA as it learns much more tighter topics that enables a more accurate prediction. To further explain this result, we refer the reader to Figure 8 where we show how LLA and thLLA segments a user history into topics. As evident from the figure, thLLA produces a fine-grained annotation of the user activities in both outdoor and desert-making themes. For instance, while LLA groups many events into "camping gear", thLLA breaks them down into camping tents and camping equipments which allows for better future prediction as the scope of future queries is reduced from all possible queries in camping gears in addition to their immediate neighboring topics (as in LLA, Figure 1) to only those queries in tents and camping equipments in addition to their immediate neighboring topics (as in thLLA, Figure 6). This happens as thLLA devotes a whole mixture to modeling camping activities and this enables learning fine-grained topics with accurate, theme-focused transitions.

5 DISCUSSIONS AND RELATED WORK

In this paper we introduced threading Latent LSTM Allocation model (thLLA) for sequence modeling. Like its predecessor LLA [25], thLLA leverages the powerful dynamic modeling capability of LSTM without blowing up the number of parameters while adding interpretability to the model. However, unlike LLA, thLLA can deal gracefully with sequence data composed of multiple themes (modes of behavior) and abrupt transitions between these modes. In addition, thLLA adds an extra layer of interpretability via its mixture components that uncovers more structure in sequence

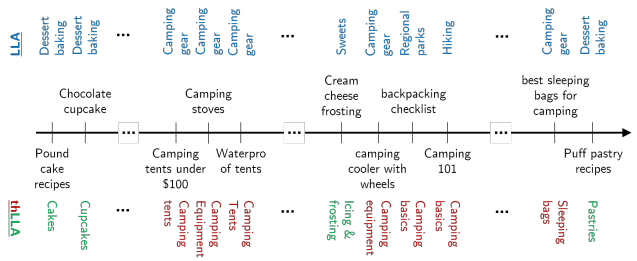


Figure 8: Illustrating how LLA and thLLA breaks a user history into topics. As evident, thLLA first breaks the history into themes: outdoor (red) and deserts (blue), then gives a more fine-grained annotation in each theme resulting in better prediction as shown in Table 1 (best viewed in color).

data. By employing a mixture of LLA models to generate each sequence, the new model threads each sequence by breaking it into smaller subsequences each of which corresponds to a given theme such as planning an outdoor activity. This threading results in easier subproblems for optimizing the dynamic LSTM model corresponding to each theme and avoids the vanishing gradient problems[13]. Experimental results over two datasets corporate our claims and proved the promise of our approach.

Our work is related to marrying topics models an deep models such as[20, 23], however, as detailed in [25], these efforts cannot capture long-range temporal dependencies which is the goal in this work. Several other line of work endow LSTM with latent variables such as [8, 11], however they focus mainly on continuous spaces such as images whereas here we focus on discreet data.

Our model is related to switching state-space (SSS) models such as [6, 12],however, here we apply this idea to recurrent models with latent variables. Furthermore, in standard SSS models all mixtures evolve at each time step and a switch variable at time t selects a them to generate the input, whereas thLLA introduces the concept of long-range jumps to break each sequence into smaller pieces thus mitigating the vanishing gradient problem [13]. Finally our model is related to attention [7], however, like LSTM, attention lack interpretability as it can't provide the extra layer of mixtures as in thLLA and in addition requires a large number of parameters that scale with the input data as opposed to the number of topics.

A APPENDIX: SMC FOR INFERENCE IN THREADING LLA MODELS

In this subsection, we discuss how to draw samples from the posterior (5), corresponding to the S-step of the stochastic EM algorithm. For sake of simplicity of notation, we consider a single user history $x_1, ..., x_T$ of length T. Thus, the poster from (5) is:

$$(y^{\star}_{1:T}, z^{\star}_{1:T}) \sim p(y_{1:T}, z_{1:T}|x_{1:T})$$
$$= \frac{\prod_t p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(x_t|z_t, \boldsymbol{\phi})p(y|\pi)}{\sum_{y_{1:T}z_{1:T}} \prod_t p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(x_t|z_t, \phi)p(y|\pi)}. \quad (9)$$

The integration and normalization can be performed efficiently, to perform this we define the following quantities which can be

computed in the standard forward pass:

$$\begin{aligned}\alpha_t &\equiv p(x_t|y_{1:t-1}, z_{1:t-1}) \\ &\propto \sum_{y_t, z_t} p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi}) \\ \gamma_t &\equiv p(y_t, z_t|z_{1:t-1}, y_{1:t-1}, x_t) \\ &\propto p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi}).\end{aligned} \tag{10}$$

Then the problem reduces to drawing from the joint posterior of $(y_{1:T}, z_{1:T})$ only given access to these *defined forward messages*.

We use an SMC algorithm to approximate the posterior (9) with point masses, i.e., weighted particles. Let $f(y_t, z_t|z_{1:t-1}, y_{1:t-1}, x_t)$ be the importance density, and P be the number of particles. We then can use SMC as in (2) with $\nu_t(z_{1:t}) = p(x_{1:t}, y_{1:t}, z_{1:t})$ (i.e. $z \leftarrow (y, z)$) being the unnormalized target distribution at time t, where the weight becomes

$$\begin{aligned}\alpha_t^p &\propto \frac{p(x_{1:t}, y_{1:t}^p, z_{1:t}^p)}{p(x_{1:t-1}, y_{1:t-1}^{a_{t-1}^p}, z_{1:t-1}^{a_{t-1}^p}) f(y_t^p, z_t^p|z_{1:t-1}^{a_{t-1}^p}, y_{1:t-1}^{a_{t-1}^p}, x_t)} \\ &= \frac{p(z_t^p|z_{1:t-1}^{a_{t-1}^p}, y_{1:t-1}^{a_{t-1}^p}, y_t^p, \omega)p(y_t^p|\pi)p(x_t|z_t^p, \boldsymbol{\phi})}{f(y_t^p, z_t^p|y_{1:t-1}^{a_{t-1}^p}, z_{1:t-1}^{a_{t-1}^p}, x_t)}.\end{aligned} \tag{11}$$

The computational cost for sampling from above at each time step is $O(M(K + H^2))$ where M is the number of mixtures, K is the number of topics (needed to compute $P(x_t|z_t)$) and H is the size of the LSTM hidden state (required to compute $P(z_t|z_{1:t-1})$, Finally M is required since we need to normalize the distribution across all possible mixture assignments for y_t.

As for the choice of the proposal distribution $f(\cdot)$, one could use the transition density $p_\omega(z_t|z_{1:t-1}, y_{1:t})p(y_t|\pi)$, in which case the algorithm is also referred to as the bootstrap particle filter. An alternative is the predictive distribution, a locally optimal proposal in terms of variance [3]:

$$\begin{aligned}&f^\star(y_t, z_t|z_{1:t-1}, y_{1:t-1}, x_t) \\ &\qquad = \frac{p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi})}{\sum_{y_t, z_t} p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi})},\end{aligned} \tag{12}$$

which is precisely one of the available forward messages:

$$\gamma_t^p = f^\star(y_t, z_t|z_{1:t-1}^{a_{t-1}^p}, y_{1:t-1}^{a_{t-1}^p}, x_t). \tag{13}$$

Notice the similarity between terms in (11) and (12). Indeed, with the choice of predictive distribution as the proposal density, the importance weight simplifies to

$$\alpha_t^p \propto \tilde{\alpha}_t^p = \sum_{y_t, z_t} p(z_t|z_{1:t-1}^{a_{t-1}^p}, y_{1:t-1}^{a_{t-1}^p}, y_t, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi}), \tag{14}$$

which is not a coincidence that the name collides with the message α_t. Interestingly, this quantity no longer depends on the current particle y_t^p, z_t^p. Instead, it marginalizes over all possible particle assignments of the current time step. This is beneficial computationally since the intermediate terms from (12) can be reused in (14). Also note that the optimal proposal relies on the fact that the normalization in (12) can be performed efficiently, otherwise the bootstrap proposal should be used. Here also cost of sampling is $O(M(K + H^2))$, as previously for (11).

Algorithm 1 Inference with Particle Gibbs

(1) Let $y_0^p = y_0$, $z_0^p = z_0$ and $\alpha_0^p = 1/P$ for $p = 1, \ldots, P$.
(2) For $t = 1, \ldots, T$:
  (a) Fix reference path: set $a_{t-1}^1 = 1$ and $y_{1:t}^1 = y_{1:t}^\star, z_{1:t}^1 = z_{1:t}^\star$ from previous iteration.
  (b) Sample ancestors $a_{t-1}^p \sim \alpha_{t-1}$ according to (11) for $p = 2, \ldots, P$.
  (c) Sample particles $y_t^p, z_t^p \sim \gamma_t^p$ according to (13) and set $y_{1:t}^p = (y_{1:t-1}^{a_{t-1}^p}, y_t^p), z_{1:t}^p = (z_{1:t-1}^{a_{t-1}^p}, z_t^p)$ for $p = 2, \ldots, P$.
  (d) Compute normalized weights α_t^p according to (14) for $p = 1, \ldots, P$.
(3) Sample $r \sim \alpha_T$, return the particle path $(y_{1:T}^{a_T^r}, z_{1:T}^{a_T^r})$.

After a full pass over the sequence, the algorithm produces Monte Carlo approximation of the posterior and the marginal likelihood:

$$\begin{aligned}\hat{p}(y_{1:T}, z_{1:T}|x_{1:T}) &= \sum_p \alpha_T^p \delta_{(y_{1:T}^p, z_{1:T}^p)}(y_{1:T}, z_{1:T}), \\ \hat{p}(x_{1:T}) &= \prod_t \frac{1}{P} \sum_p \tilde{\alpha}_t^p.\end{aligned} \tag{15}$$

where δ is the Dirac-delta function. The inference is completed by a final draw from the approximate posterior,

$$y_{1:T}^\star, z_{1:T}^\star \sim \hat{p}(y_{1:T}, z_{1:T}|x_{1:T}), \tag{16}$$

which is essentially sampling a particle path indexed by the last particle. Specifically, the last particle y_T^p, z_T^p is chosen according to the final weights α_T, and then earlier particles can be obtained by tracing backwards to the beginning of the sequence according to the ancestry indicators a_t^p at each position.

However, as the length of the sequence T increases, the number of particles needed to provide a good approximation grows exponentially. This is the well-known depletion problem of SMC [3].

One elegant way to avoid simulating enormous number of particles is to marry the idea of MCMC with SMC [3]. The idea of such Particle MCMC (PMCMC) methods is to treat the particle estimate $\hat{p}(\cdot)$ as a proposal, and design a Markov kernel that leaves the target distribution invariant. Since the invariance is ensured by the MCMC, it does not demand SMC to provide an accurate approximation to the true distribution, but only to give samples that are approximately distributed according to the target. As a result, for any fixed $P > 0$ the PMCMC methods ensure the target distribution is invariant.

We choose the Gibbs kernel that requires minimal modification from the basic SMC. The resulting algorithm is Particle Gibbs (PG), which is a conditional SMC update in a sense that a reference path $y_{1:T}^{\text{ref}}, z_{1:T}^{\text{ref}}$ with its ancestral lineage is fixed throughout the particle propagation of SMC. It can be shown that this simple modification to SMC produces a transition kernel that is not only invariant, but also ergodic under mild assumptions. In practice, we use the assignments from previous step as the reference path. The final algorithm is summarized in Algorithm 1.

REFERENCES

[1] M. Aly, A. Hatch, V. Josifovski, and V.K. Narayanan. 2012. Web-scale user modeling for targeting. In *Conference on World Wide Web*. ACM, 3–12.

[2] Mark Andrews and Gabriella Vigliocco. 2010. The hidden Markov topic model: A probabilistic model of semantic representation. *Topics in Cognitive Science* 2, 1 (2010), 101–113.

[3] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. 2010. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (2010).

[4] David Blei and Peter I. Frazier. 2011. *Distance Dependent Chinese Restaurant Processes*. JMLR.

[5] D. Blei, A. Ng, and M. Jordan. 2002. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). MIT Press, Cambridge, MA.

[6] Timothy W Cacciatore and Steven J Nowlan. 1994. Mixtures of controllers for jump linear and non-linear plants. In *Advances in neural information processing systems*. 719–726.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*. 2980–2988.

[9] Arnaud Doucet, Nando de Freitas, and Neil Gordon. 2001. *Sequential Monte Carlo Methods in Practice*. Springer.

[10] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press.

[11] Mevlana Gemici, Chia-Chun Hung, Adam Santoro, Greg Wayne, Shakir Mohamed, Danilo J Rezende, David Amos, and Timothy Lillicrap. 2017. Generative Temporal Models with Memory. *arXiv preprint arXiv:1702.04649* (2017).

[12] Zoubin Ghahramani and Geoffrey E. Hinton. 1996. *Parameter estimation for linear dynamical systems*. Technical Report.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Diane J Hu, Rob Hall, and Josh Attenberg. 2014. Style in the long tail: Discovering unique interests with latent variable models in large scale social e-commerce. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1640–1649.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv* (2016).

[16] Mirjam Köck and Alexandros Paramythis. 2011. Activity sequence modelling and dynamic clustering for personalized e-learning. *User Modeling and User-Adapted Interaction* 21, 1-2 (2011), 51–97.

[17] Mandy Korpusik, Shigeyuki Sakaki, and Francine Chen Yan-Ying Chen. 2016. Recurrent Neural Networks for Customer Purchase Prediction on Twitter. *CBRecSys 2016* (2016), 47.

[18] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv* (2015).

[19] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.

[20] Christopher E Moody. 2016. Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *arXiv preprint arXiv:1605.02019* (2016).

[21] Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv preprint arXiv:1606.07461* (2016).

[22] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.

[23] Fei Tian, Bin Gao, and Tie-Yan Liu. 2016. Sentence Level Recurrent Topic Model: Letting Topics Speak for Themselves. *arXiv preprint arXiv:1604.02038* (2016).

[24] Martin J Wainwright and Michael I Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1, 1-2 (2008), 1–305.

[25] Manzil Zaheer, Amr Ahmed, and Alex J. Smola. 2017. Latent LSTM Allocation: Joint Clustering and Non-Linear Dynamic Modeling of Sequence Data. In *ICML*.

[26] Manzil Zaheer, Michael Wick, Jean-Baptiste Tristan, Alex Smola, and Guy L Steele Jr. 2016. Exponential stochastic cellular automata for massively parallel inference. In *AISTATS: 19th Intl. Conf. Artificial Intelligence and Statistics*.