Latent LSTM Allocation
Joint Clustering and Non-Linear Dynamic Modeling of Sequential Data

Manzil Zaheer [1] **Amr Ahmed** [2] **Alexander J Smola** [1]

Abstract

Recurrent neural networks, such as long-short term memory (LSTM) networks, are powerful tools for modeling sequential data like user browsing history (Tan et al., 2016; Korpusik et al., 2016) or natural language text (Mikolov et al., 2010). However, to generalize across different user types, LSTMs require a large number of parameters, notwithstanding the simplicity of the underlying dynamics, rendering it uninterpretable, which is highly undesirable in user modeling. The increase in complexity and parameters arises due to a large action space in which many of the actions have similar intent or topic. In this paper, we introduce Latent LSTM Allocation (LLA) for user modeling combining hierarchical Bayesian models with LSTMs. In LLA, each user is modeled as a sequence of actions, and the model jointly groups actions into topics and learns the temporal dynamics over the topic sequence, instead of action space directly. This leads to a model that is highly interpretable, concise, and can capture intricate dynamics. We present an efficient Stochastic EM inference algorithm for our model that scales to millions of users/documents. Our experimental evaluations show that the proposed model compares favorably with several state-of-the-art baselines.

1. Introduction

Sequential data prediction is an important problem in machine learning spanning over a diverse set of applications ranging from text (Mikolov et al., 2010) to user behavior (Köck & Paramythis, 2011). For example, when applied to statistical language modeling, the goal is to predict the next word in textual data given context, very similar to that in user activity modeling where the aim is to predict the next activity of the user given the history. Accurate user activity modeling is very important for serving relevant, personalized, and useful contents to the user. A good model of sequential data should be accurate, sparse, and interpretable. Unfortunately, none of the existing techniques for user or language modeling satisfy all of these requirements.

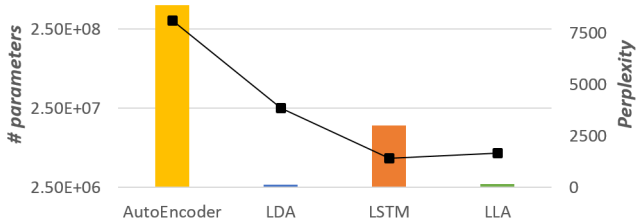


Figure 1. LLA has better perplexity (lower is better) than LDA but much fewer parameters than LSTMs, as shown in a language modeling task on Wikipedia.

The state-of-the-art for modeling sequential data is to employ recurrent neural networks (RNN) (Lipton et al., 2015), such as LSTMs (Long-Short Term Memory) (Hochreiter & Schmidhuber, 1997). Such RNNs have been shown to be effective at capturing long and short patterns in data, e.g. token-level semantic as well as syntactic regularities in language (Jozefowicz et al., 2016). However, the neural network representations are generally uninterpretable and inaccessible to humans (Strobelt et al., 2016). Moreover, the number of parameters of the model is proportional to the number of observed word types or action types, which can grow to tens or hundreds of millions. Note that for user modeling task, character level RNN is not feasible because user actions are often not words but hash indices or URLs.

On the other hand of the spectrum, latent variable models with multi-task learning, such as LDA (Blei et al., 2002) and other topic model variants, which are strictly not sequence models, proved to be powerful tools for uncovering latent structure in both text and user data (Aly et al., 2012) with good commercial success (Hu et al., 2014). Topic models are popular for their ability to organize the data into a smaller set of prominent themes or topics through statistical strength sharing across users or documents. These topic representations are generally accessible to humans and easily lend themselves to being interpreted.

In this paper, we propose Latent LSTM Allocation (LLA), a model that bridges the gap between the sequential RNN's and the non-sequential LDA. LLA borrows graph-

[1]Carnegie Mellon University, Pittsburgh PA – work done while at Google [2]Google Inc, Mountain View CA. Correspondence to: Manzil Zaheer <manzil@cmu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

ical model techniques to infer topics (groups of related word or user activities) by sharing statistical strength across users/documents and recurrent deep networks to model the dynamics of topic evolution inside each sequence (document or user activities) rather than at user action/word level (Sec. 3.1). LLA inherits sparsity and interpretability from LDA, while borrowing accuracy from LSTM. We provide various variants of LLA that trade model size vs. accuracy without sacrificing interpretability (Sec. 3.3). As shown in Figure 1, for the task of language modeling on the Wikipedia dataset, LLA achieves comparable accuracy to LSTM while being as sparse as LDA in terms of models size. We give an efficient inference algorithm for parameter inference in LLA (Sec. 3.2) and show is efficacy and *interpretability* over several datasets (Sec. 4).

2. Background

In this section, we provide a brief review of user/language modeling and LSTMs.

2.1. User/Language Modeling

User activity modeling and language modeling amounts to learning a function that computes the log probability, $\log p(\boldsymbol{w}|\text{model})$, of a user activity or sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$. Subsequently, this function can be used to predict the next set of actions or words. This function can be decomposed and learned in different ways under various assumptions. Imposing a bag of words assumption - as used in LDA - leads to ignoring the sequence information and yields $\sum_{i=1}^{n} \log p(w_i|\text{model})$. Alternatively, one could decompose according to the chain rule into sum of the conditional log probabilities $\sum_{i=1}^{n} \log p(w_i \mid w_1, \ldots, w_{i-1}, \text{model})$, thereby preserving temporal information and use some RNN to model $\log p(w_i \mid w_1, \ldots, w_{i-1}, \text{model})$ (Mikolov et al., 2010; Sundermeyer et al., 2012).

2.2. Long Short-Term Memories

Temporal aspect is very important for user activity modeling. LSTM, a type of RNN, is well suited for the task as it can learn from experience to classify, process, and predict time series when there are very long time lags of unknown size between important events. LSTM is designed to cope with the vanishing gradient problem inherent in simple RNNs (Hochreiter & Schmidhuber, 1997). This is one of the main reasons why LSTMs outperform simple RNNs, hidden Markov models, and other sequence learning methods in numerous application.

In general, a RNN is a triplet (Σ, S, δ):

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

RNN maintains an internal state and at each time step take an input $\mathbf{x}_t$ and updates its state $\mathbf{s}_t$ by applying the transition function made up of neural network δ the previous time step's state $\mathbf{s}_{t-1}$ and the input.

Often the input is not available directly as elements of Σ and the output desired is not the state of the RNN. In such cases, input is appropriately transformed and desired output is produced at each time step from the state $\mathbf{s}_t$:

$$\mathbf{y}_t = g(\mathbf{s}_t),$$

where g is an arbitrary differentiable function.

For example, in a regular recurrent language model (RRLM) (Mikolov et al., 2010) a document is treated as a sequence and an LSTM is trained to predict directly the next word conditioned on the sequence of words before it, i.e. maximize $p(w_t|w_{t-1|}, w_{t-2}, ..., w_0; \text{model})$. In this case, the input transformation is done by using a word lookup table from word to a vector in Σ. This word representation is used to update the state of the LSTM. The output transformation is the projection of $\mathbf{s}_t$ into a vector of size of the vocabulary V followed by a softmax. However, this method will require two large matrices of dimension $V \times H$ and cannot handle out of vocabulary words.

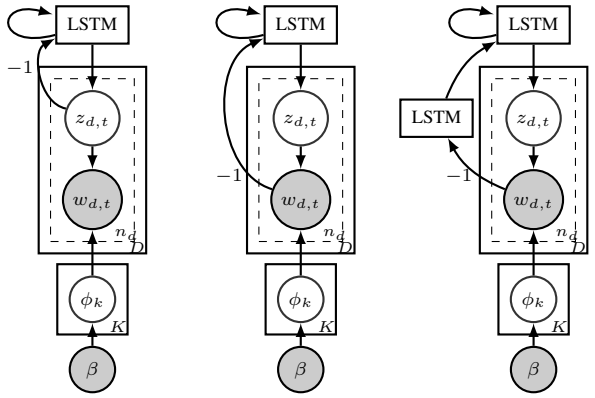
To overcome above mentioned shortcomings, another input transformation has been proposed, which looks at characters directly instead of words (Ling et al., 2015). On the spelling of the words another LSTM is run having characters as the input and the final state is used as the word representation. In this case, the memory requirement is reduced to half and the model can handle out-of-vocabulary words like typographical errors or new words made from composing existing ones. To use character level embedding, we can simply replace the word lookup table with the representation obtained from character-level LSTM. In case of natural languages, a similar trick of using a character-level LSTM to emit words can be applied as the output transformation as well. However, user modeling outputs (hash indices and URLs) lack morphological structure, and hence cannot leverage the technique.

3. Latent LSTM Allocation

In this section, we provide a detailed description of the proposed LLA model and its variant for performing joint clustering and modeling non-linear dynamics. An ideal model for such a task should have three characteristics. First, it should have a low number of parameters. Second, it should be interpretable, allowing human analysis of the components. Lastly and most importantly, the model should be accurate in terms of predicting future events. We show how LLA satisfies all of these requirements.

3.1. Generative model

In this model, we try to marry LDA with its strong interpretability capabilities with LSTM having excellent track record in capturing temporal information. In this regard,



(a) Topic LLA (b) Word LLA (c) Char LLA

Figure 2. Graphical models for LDA and variants of proposed latent LSTM Allocation (LLA). In a slight abuse of plate notation, we do not imply exchangibility by the dashed plate diagram and -1 on an edge means the dependence is from one time step back.

we propose a factored model, i.e. we use the LSTM to model sequence of topics $p(z_t|z_{t-1}, z_{t-2}, ..., z_1)$ and multinomial-Dirichlet to model word emissions $p(w_i|z_i)$, similar to LDA. Suppose we have K topics, vocabulary size V, and a document collection $\mathcal{D}$ where each document $d \in \mathcal{D}$ is composed of N_d words. With these notations, the complete generative process can be illustrated as in Figure 2a and can be described as:

1. for $k = 1$ to K
   (a) Choose topic $\boldsymbol{\phi}_k \sim \mathsf{Dir}(\beta)$
2. for each document d in corpus $\mathcal{D}$
   (a) Initialize LSTM with $\mathbf{s}_0 = 0$
   (b) for each word index t from 1 to N_d
      i. Update $\mathbf{s}_t = \text{LSTM}(z_{d,t-1}, \mathbf{s}_{t-1})$
      ii. Get topic proportions at time t from the LSTM state, $\boldsymbol{\theta} = \mathsf{softmax}_K(\mathbf{W}_p \mathbf{s}_t + \mathbf{b}_p)$
      iii. Choose a topic $z_{d,t} \sim \mathsf{Categorical}(\boldsymbol{\theta})$
      iv. Choose word $w_{d,t} \sim \mathsf{Categorical}(\boldsymbol{\phi}_{z_{d,t}})$

Under this model, the marginal probability of observing a given document d can be written as:

$$p(\boldsymbol{w}_d|\text{LSTM}, \boldsymbol{\phi}) = \sum_{\boldsymbol{z}_d} p(\boldsymbol{w}_d, \boldsymbol{z}_d|\text{LSTM}, \boldsymbol{\phi})$$
$$= \sum_{\boldsymbol{z}_d} \prod_t p(w_{d,t}|z_{d,t}; \boldsymbol{\phi}) p(z_{d,t}|\boldsymbol{z}_{d,1:t-1}; \text{LSTM}).$$

Here $p(z_{d,t}|\boldsymbol{z}_{d,1:t-1}; \text{LSTM})$ is the probability of generating topic for the next word in the document given topics of previous words and $p(w_{d,t}|z_{d,t}; \phi)$ is the probability of generating word given the topic, illustrating the simple modification of LSTM and LDA based language models.

The advantage of this modification is two fold. Firstly, we obtain a factored model as shown in Figure 3, thereby the number of parameters is reduced significantly compared to RRLM. This is because, unlike RRLM we operate at topic level instead of words directly and the number of topics is much smaller than the vocabulary size, i.e., $K << V$. This allows us to get rid of large $V \times H$ word embedding look-up table and $V \times H$ matrix used in softmax over the entire vocabulary. Instead we map from hidden to state to topics first using a $K \times H$ matrix, which will be dense and then from topics to words using a $V \times K$ matrix under the Dirichlet-multinomial model similar to LDA which will be very sparse. Secondly, this model is highly interpretable. We recover global themes present in the documents as ϕ. The LSTM output represents topic proportions for document/user at time t. The LSTM input over topics can capture semantic notion of topics.

3.2. Inference

As the computation of the true posterior of LLA as described above is intractable, we have to resort to an approximate inference technique like mean field variational inference (Wainwright & Jordan, 2008) or stochastic EM (SEM) (Zaheer et al., 2016). Below we describe the generic aspects of the inference algorithm for LLA. Model specific aspects are relegated to the appendix. [1]

Given a document collection, the inference task is to find the LSTM weights and word|topic probability matrix ϕ. This can be carried out using SEM. We begin by writing out the likelihood and lower bounding it as,

$$\sum_d \log p(\boldsymbol{w}_d|\text{LSTM}, \boldsymbol{\phi})$$
$$\geq \sum_d \sum_{\boldsymbol{z}_d} q(\boldsymbol{z}_d) \log \frac{p(\boldsymbol{z}_d; \text{LSTM}) \prod_t p(w_{d,t}|z_{d,t}; \boldsymbol{\phi})}{q(\boldsymbol{z}_d)}, \tag{1}$$

for any distribution q. Then the goal is to ensure an increase in this evidence lower bound (ELBO) with each iteration in expectation. Following the suit of most EM based inference methods, each iteration involves two phases, each of which is described below:

SE-step: For fixed LSTM parameters and $\boldsymbol{\phi}$, we sample the topic assignments $\boldsymbol{z}$. This acts as an unbiased sample estimate of the expectation with respect to the conditional distribution of $\boldsymbol{z}$ required in traditional EM. This sampled estimate not only enjoys similar statistical convergence guarantees (Nielsen, 2000) but also provides many computational benefits like speed-up and reduced memory bandwidth requirements (Zaheer et al., 2016).

Under LLA, the conditional probability for topic at time step t given word at time t and past history of topics is,

$$p(z_{d,t} = k|w_{d,t}, z_{d,1:t-1}; \text{LSTM}, \boldsymbol{\phi})$$
$$\propto p(z_{d,t} = k|z_{d,1:t-1}; \text{LSTM}) p(w_{d,t}|z_{d.t} = k; \boldsymbol{\phi}) \tag{2}$$

The first term represents probability of various topics predicted by LSTM dynamics after final softmax and second term comes from multinomial-Dirichlet word emission model given the topic. Naïvely sampling from this distribution would cost $O(KH + H^2)$ per word.

[1] Available at http://manzil.ml/lla.html

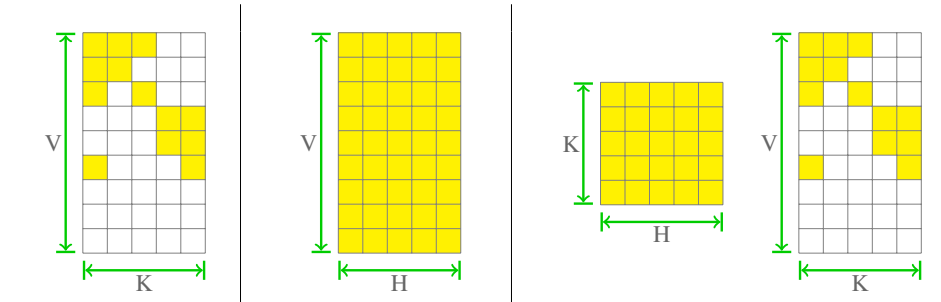


Figure 3. Different parameters employed by LDA, LSTM and LLA. K is number of topics, V, is vocabulary size, and H is the dimension of LSTM state. (a) Topics of LDA, (b) word embedding of LSTM (c) factored topic embedding of LLA.

Optionally, one could speed up this sampling. Note that the second term in (2) has a sparse structure. For sampling from (2) the computation of the normalizer is not essential. To elaborate, let us define n_{wk} as the number of times word w currently assigned to topic k and n_k as the number of tokens assigned to topic k. Explicitly writing down first term:

$$p(w_{d,t} = w|z_{d,t} = k, \boldsymbol{\phi}) = \phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta} = \underbrace{\frac{n_{wk}}{n_k + V\beta}}_{\text{sparse}} + \underbrace{\frac{\beta}{n_k + V\beta}}_{\text{dense}} \quad (3)$$

There is an inherent sparsity present in n_{wk}, as a given word would be typically about only a handful of topics, $K_w \ll K$. The second term represents the global count of tokens in the topic and will be dense, regardless of the number of documents, words or topics.

Following (Li et al., 2014), we devise an *optional* fast sampling strategy for exploiting this sparsity and construct a Metropolis-Hastings sampler. The idea is to replace the low weight dense term by a cheap approximation, while keeping the high weight sparse term exact. This leads to a proposal distribution that is close to (2), while at the same time allowing us to draw from it efficiently:

- First draw a biased coin to decide whether to draw from the sparse n_{wk} term or from the dense term. The bias is
  $$b = \frac{p}{p+q}, \text{ where}$$
  $$p = \sum_{k:n_{wk}\neq 0} \frac{n_{wk}}{n_k + V\beta} p(z_{d,t} = k|\boldsymbol{z}_{d,1:t-1}; \text{LSTM})$$
  $$q = \sum_k \frac{\beta}{n_k + V\beta} \qquad \text{(pre-computed)}.$$
- If we draw from the sparse term, the cost is $O(K_w H + H^2)$, else the cost is $O(H^2)$ using the alias trick.
- Finally, perform a MH accept/reject move by comparing the approximation with the true distribution.

M-step: For fixed topic assignment $\boldsymbol{z}$, update the $\boldsymbol{\phi}$ and LSTM so as to improve the likelihood in expectation. As the dependence of likelihood on LSTM parameters and $\boldsymbol{\phi}$ are independent given $\boldsymbol{z}$, we can update them independently and in parallel. For the former, we use the closed form expression for the maximizer,

$$\phi_{wk} = \frac{\#\{w_{d,t} = w \text{ and } z_{d,t} = k\} + \beta}{\#\{n_{wk} = k\} + V\beta} = \frac{n_{wk} + \beta}{n_k + V\beta},$$

and for the latter we resort to stochastic gradient ascent which will increase the likelihood in expectation.

The execution of the whole inference and learning process includes several iterations involving the above mentioned two phases as outlined in Algorithm 1. This inference procedure is not an ad-hoc method, but an instantiation of the well studied SEM method. We ensure that in each iteration we increase the ELBO in expectation. Also, it is just a coincidence that for all such latent variable models the equations for SE-step looks like Gibbs sampling. (Exploiting this coincidence, in fact, one can prove that parallel Gibbs update will work for such models, cf (Tassarotti & Steele Jr, 2015; Zaheer et al., 2016) whereas in general parallel Gibbs sampling fails miserably e.g. for Ising models.) Moreover, we found empirically that augmenting the SEM inference with a beam search improved the performance.

Automatic differentiation software packages such as TensorFlow (Abadi et al., 2015) significantly speed up development time, but have limited control structures and indexing. Whereas random sampling requires varied control statements, thus was implemented separately on the CPU. Furthermore, the SEM saved us precious GPU-CPU bandwidth by transmitting only an integer per token instead of a K-dimensional variational parameter.

3.3. Adding Context

Utilizing the word or user action, $w_{d,t}$, directly would be more informative to model the dynamics and predict the next topic. For example, rather than only knowing the previous action belonged to "electronics purchase" topic, knowing exactly that a camera was bought, makes it easier to predict user's next interest, e.g. camera lenses.

Algorithm 1 Stochastic EM for LLA

Input: Document corpus $\mathcal{D}$.

Initialize ϕ and LSTM with a few iterations of LDA
repeat
  SE-Step:
    for all document $d \in \mathcal{D}$ **in parallel do**
      for $t \leftarrow 1$ to N_d (possibly with padding) **do**
        $\forall k \in \{1, \cdots, K\}$, i.e., for every topic index obtain by LSTM forward pass: $\pi_k = \phi_{w_{d,t}k} p(z_{d,t} = k | \boldsymbol{z}_{d,1:t-1}; \text{LSTM})$.
        Sample $z_{d,t} \sim \mathsf{Categorical}(\boldsymbol{\pi})$
      end for
    end for
  M-Step:
    Collect sufficient statistics to obtain:

$$\phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta}, \quad \forall w, k$$

    for mini-batch of documents $\mathcal{B} \subset \mathcal{D}$ **do**
      Compute the gradient by LSTM backward pass

$$\frac{\partial \mathcal{L}}{\partial \text{LSTM}} = \sum_{d \in \mathcal{B}} \sum_{t=1}^{N_d} \frac{\partial \log p(z_{d,t} | z_{d,1:t-1}; \text{LSTM})}{\partial \text{LSTM}}$$

      Update LSTM parameters by stochastic gradient descent methods such as Adam (Kingma & Ba, 2014).
    end for
until Convergence

In order to incorporate the exact context, we construct a variant of LLA, called **word LLA**, where the LSTM is provided with an embedding of previous word $w_{d,t-1}$ directly instead of $z_{d,t-1}$. The corresponding graphical model is shown in Figure 2b and the joint likelihood is:

$$\sum_d \log p(\boldsymbol{w}_d | \text{LSTM}, \phi)$$
$$= \sum_d \sum_t \log \sum_{z_{d,t}} p(w_{d,t} | z_{d,t}; \phi) p(z_{d,t} | \boldsymbol{w}_{d,1:t-1}; \text{LSTM})$$

A SEM inference strategy can be devised similar to that of topic LLA as presented in section 3.2.

However, this modification brings back the model to the dense and high number of parameter regime as a large trainable $V \times H$ lookup table for the word embeddings is needed for the input transformation. This problem can be mitigated by using char-level LSTM (**char LSTM** for short) to construct the input transformation. Such character-level LSTM have recently shown promising results in generating structured text (Karpathy et al., 2015) as well as in producing semantically valid word embeddings with a model we will refer to as char-LSTM) (Ling et al., 2015). The character-level models do not need the huge lookup table for words, as they operate directly on the characters and the number of distinct characters is extremely small. We chose the latter as our input transformation and briefly describe it below.

The input word w is decomposed into a sequence of characters $c_1, \ldots, c_m$, where m is the length of w. Each character c_i is transformed using a lookup table and fed into a bidirectional LSTM The forward LSTM evolves on the character sequence and produces a final state $\mathbf{h}_m^f$. While the backward LSTM receives as input the reverse sequence, and yields its own final state $\mathbf{h}_0^b$. Both LSTMs use a different set of parameters. The representation $\mathbf{e}_w^c$ of word w is obtained by combining forward and backward final states:

$$\mathbf{e}_w^C = \mathbf{D}^f \mathbf{s}_m^f + \mathbf{D}^b \mathbf{s}_0^b + \mathbf{b}_d, \tag{4}$$

where $\mathbf{D}^f$, $\mathbf{D}^b$ and $\mathbf{b}_d$ are parameters that determine how the states are combined. This e_w^c is provided to the topic level LSTM as the input providing information about the word. Thus we are able to incorporate more context information by providing some information about the previous word without the need to have great number of parameters. We call this model, **char LLA**.

Model	Input Tranformation	Output Transformation
word LSTM	Word Embedding	Softmax over vocabulary
Char LSTM	Char Embedding	Softmax over vocabulary
Topic LLA	Topic Embedding	Topic based
Word LLA	Word Embedding	Topic based
Char LLA	Char Embedding	Topic based

Table 1. Enumerating different input and output representation, leading to variants of LLA models. Note we exclude char level output transformation due to its inapplicability to user modeling.

The different variants of LLA and LSTM as language model can be unified and thought of having different input and out transformations over LSTM for capturing the dynamics. The possible combinations are listed in Table 1. We will study each of them empirically next.

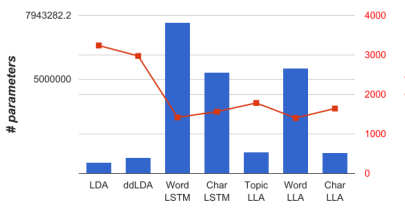
4. Experimental Results

We perform a comprehensive evaluation of our model against several deep models, dynamic models, and topic models. *For reproducibility we focus on the task of language modeling over the publicly available Wikipedia dataset*, and for generality, we show additional experiments on the less-structured domain of user modeling.

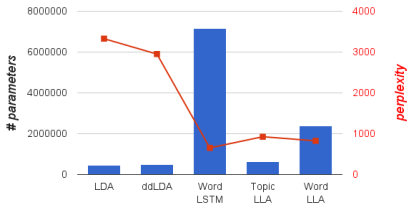
4.1. Setup

For all experiments we follow the standard setup for evaluating temporal models, i.e. divide each document (user history) into 60% for training and 40% for testing. The task is to predict the remaining 40% of the document (user data) based on the representation inferred from the first 60% of the document. We use perplexity as our metric (lower values are better) and compare our models (topic-LLA, word-LLA, and char-LLA) against the following baselines:

- **Autoencoder:** A feed forward neural network that comprises of encoder and decoder. The encoder projects the input data into a low-dimensional representation and the decoder reconstructs the input from this representation. The model is trained to minimize reconstruction error.
- **LSTMs:** We consider both word-level LSTM language model (word-LSTM) and character-level hierarchical LSTM (char-LSTM) language model.

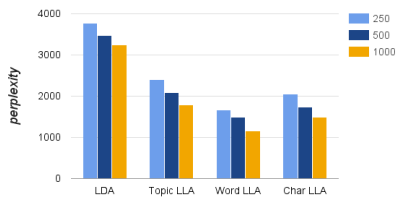


(a) Wikipedia



(b) User search click history

Figure 4. Test perplexity and number of parameters of various models. Bars represent model sizes and the solid curve represents perplexity over testset (lower is better).



Wikipedia

Figure 5. The effect of the number of topics over the performance of LDA and LLA.

- **LDA:** The vanilla version trained using collapsed Gibbs sampling over the bag of word representation.
- **ddLDA:** Distance-dependent LDA is a variant of LDA incorporating the word sequence. It is based on a fixed-dimensional approximaiton of the distance-dependent Dirichlet process (Blei & Frazier, 2011). In this model a word is assigned to a topic based on the popularity of the topics assigned to nearby words. Note that this model subsumes other LDA-based temporal models such as hidden Markov topic Model (Andrews & Vigliocco, 2010) and RCRP based models (Ahmed & Xing, 2008).

We trained all deep models using stochastic gradient decent with Adam (Kingma & Ba, 2014). All LSTM and LLA based models used the sampled softmax method for efficiency (Cho et al., 2015). Finally, all hyper-parameters of the models were tuned over a development set.

4.2. Language modeling

We used the publicly available Wikipedia dataset to evaluate the models on the task of sequence prediction. Extremely short documents, i.e. documents having length less than 500 words were excluded. The dataset contains ~0.7 billion tokens and ~1 million documents. The most frequent 50k word-types were retained as our vocabulary. Unless otherwise stated, we used 1000 topics for LLA and LDA variants. For LSTM and LLA variants, we selected the dimensions of the input embedding (word or topic) and evolving latent state (over words or topics) in the range of $\{50, 150, 250\}$. In case of character-based models, we tuned the dimensions of the character embedding and latent state (over characters) in the range of $\{50, 100, 150\}$.

Accuracy vs model size Figure 4(a) compares model size in terms of number of parameters with model accuracy in terms of test perplexity. As shown in the figure, LDA yields the smallest model size due to the sparsity bias implicit in the Dirichlet prior over the topic|word distributions. On the other hand, word-LSTM gives the best accuracy due to its ability to utilize word level statistics; however, the model size is order of magnitudes larger than LDA. Char-LSTM achieves almost 50% reduction in model size over word-LSTM at the expense of a slightly worse perplexity. The figure also shows that LLA variants (topic-LLA, word-LLA, and char-LLA) can trade accuracy vs model size while still maintaining interpretability since the output of the LSTM component is always at the topic level. Note that the figure depicts the smallest word-LSTM at this perplexity level, as our goal is not to beat LSTM in terms of accuracy, but rather to provide a tuning mechanism that can trade-off perplexity vs model size while still maintaining interpretability. Finally, compared to LDA, which is a widely used tool, LLA variants achieve a significant perplexity reduction at a modest increase in model size while still maintaining topic sparsity. As shown in Figure 1, the autoencoder model performs poorly in terms of model size and perplexity thus we eliminated it from Figure 4(a) and the following figures to avoid cluttering the display.

Convergence over time At first glance, LLA seems to be more involved than both LDA and LSTM. So, we raise the question whether the added complexity leads to a slower training. Figure 8 shows that compared to LSTM based models, LLA variants achieve comparable convergence speed. Moreover, compared to fast LDA samplers (Zaheer et al., 2017), our LLA variants introduce only a modest increase in training time. The figure also shows that character based models (char-LSTM and char-LLA) are slightly slower to train compared to word level variants due to their nested nature and the need to propagate gradients over both word and character level LSTMs.

Ablation study Since both LDA and LLA result in interpretable models, we want to explore if LDA can achieve a perplexity similar to a given LLA model by just increasing the number of topics in LDA. Figure 5 shows the performance of LDA and variants of LLA for different number of topics. As can be seen from the figure, even with 250 topics, all LLA based models achieve much lower perplexity than LDA with 1000 topics. In other words, LLA improves over LDA not because it uses a slightly larger model, but rather because it models the sequential order in the data.

Interpretability Last but not least, we demonstrate the interpretability of our models. Similar to LDA, the proposed LLA also uncovers global themes, a.k.a topics prevalent in the dataset. We found qualitatively the topics produced by LLA to be cleaner. For example, in Table 2 we show a topic about funding, charity, and campaigns recovered by both. LDA includes spurious words like Iowa in the topic just because it co-occurs in the same documents. Whereas modeling the temporal aspect allows LLA to correctly switch topics at different parts of the sentence producing cleaner topic regarding the same subject.

LDA	foundation, **iowa**, charity, fund, money, campaign, raised, donated, funds, donations, raise, support, charitable, **million**, donation
LLA	fund, foundation, money, funds, support, charity, funding, donations, campaign, raised, donated, **trust**, raising, **contributions**, **awareness**

Table 2. Cleaner topics produced by LLA vs LDA

Modeling based on only co-occurrences in LDA leads to further issues. For example, strikes among mine workers are common, so the two words will co-occur heavily but it does not imply that strikes and mining should be in the same topic. LDA assign both the words in the same topic as shown in Table 3. Modeling sentences as an ordered sequence allows distinction between the subjects and objects in the sentence. In the previous example, this leads to factoring into two separate topics of strikes and mine workers.

The topic LLA provides embedding of the topics in a Euclidean metric space. This embedding allows us to understand the temporal structure learned by the model: topics close in this space are expected to appear in close sequential proximity in documents. To understand this, we built a topic hierarchy using the embeddings which uncovers interesting facts about the data. For example in Figure 6, we show a portion of the topic hierarchy discovered by topic-LLA with 1000 topics. For each topic we list the top few words. The theme of the figure is countries in Asia. It groups topics relating to one country together and puts topics related to neighboring countries close by. Three prominent clusters are shown form top to bottom which corresponds to moving from west to east on the map of Asia. Top cluster is about Arab world, second one represent the Indian sub-continent, and the third one starts with south-east Asia like Philippines. (The topic abs gma cbn represents TV station in south-east Asia.) The complete hierarchy of topic is very meaningful and can be viewed at `http://manzil.ml/lla.html`.

4.3. User modeling

We use an anonymized sample of user search click history to measure the accuracy of different models on predicting users' future clicks. An accurate model would enable better user experience by presenting the user with relevant content. The dataset is anonymized by removing all items appearing less than a given threshold, this results in a dataset of ~50 million tokens, and 40K vocabulary size. This domain is less structured than the language modeling task since users' click patterns are less predictable than the sequence of words which follow definite syntactic rules. We used a setup similar to the one used in the experiments over the Wikipedia dataset for parameters ranges and selections.

Figure 4(b) gives the same conclusion as Figure 4(a): LLA variants achieve significantly lower perplexity compared to LDA and at the same time they are considerably smaller models compared to LSTM. Furthermore, even compared

LDA	strike, strikes, striking, miners, strikers, workers workers, day, began, general, called, pinkerton, action, hour, hunger, keefe
LLA	union, unions, strike, workers, labor, federation, trade, afl, bargaining, cio, organization, relations, strikes, national, industrial
	mining, coal, mine, mines, gold, ore, miners, copper, iron, rush, silver, mineral, deposits, minerals, mined

Table 3. Factored topics produced by LLA vs LDA

to ddLDA, an improved variant of LDA, our proposed LLA achieves better perplexity. ddLDA models time by introducing smoothness constraints that bias future actions to be generated from recent topics; however, it does not possess a *predictive* action model. As a result, it can neither model the fact that "booking a flight" topic should not be repeated over a short course of time nor that "booking a hotel" topic would likely follow shortly, but not necessarily immediately, after "booking a flight" topic. LLA, on the other hand, is capable of capturing this intricate dynamics by modeling the evolution of user topics via an LSTM – an extremely powerful dynamic model. This point is more evident if we consider the following *hand-crafted*[2] user click trace in context of the topics depicted in Figure 7:

theknot.com zola.com weddingwire.com www.bridalguide.com pinterest.com food.com doityourself.com .....

There are four topics represented and color-coded (best viewed in color): wedding (sixth topic from top), social media (fourth from top), food (third form bottom) and home improvement (second form top) – all in Figure 7. We asked each model to predict what would be the three most likely topics that would appear next in current user's session. LDA predicted wedding as the top topic followed by a tie for the remaining topics. ddLDA, whose output is based on exponential decay, yields wedding as most likely, followed by doityourself topic, and then the food topic as expected. In contrast, LLA ranks the most likely three topics as: doityourself topic, houzz topic (top topic in Figure 7), and the wedding topic. This is indicative of LLA learning that once a user starts in the doitourself topic, the user will stay longer in this part of the tree for the task and wander in the nearby topics for a while. Moreover, LLA still remembers the wedding topic, and unlike other models, LLA did not erroneously pick neighbors of the wedding topic among the three most likely topics to follow.

Finally, to demonstrate the interpretability of our model, in Figure 7, we show a portion of the topic hierarchy discovered by topic-LLA with 250 topics. For each topic we list the top few clicked items. The theme of the figure is wedding and various house chores. Three prominent clusters are shown from top to bottom. The top cluster is about house renovations and wedding. Second cluster captures

[2]For privacy concerns, we construct an artificial example manually for illustration purposes.

Figure 6. Topic embedding discovered form the Wikipedia dataset

Figure 7. Topic embedding discovered form the user search click history

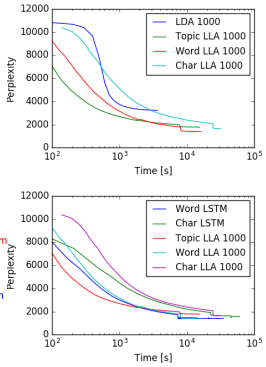


Figure 8. Convergence speed for various models.

makeup, designer shoes, and fashion. The bottom cluster capture kitchen, cooking and gardening. It is clear form the hierarchy that LLA groups topics into the embedding space based on *sequential proximity* in the search click history.

4.4. Effect of Joint Training

One might wonder what would happen if we first train LDA and then simply train LSTM over the topic assignments from the LDA. We refer to this baseline as "independent learner" since LDA and LSTM are trained independently. In Table 4, we compare its performance against LLA, which jointly learns the topics and the evolution dynamics. As we can see, joint training is significantly better, since the LSTM will fit the random errors introduced by the topic assignments inferred form the LDA model, and in fact LSTM will learn to be as good as the sequence produced by an LDA model (which is time-oblivious to start with). The effect is more pronounced in the user history data due to the unstructured nature of this domain.

Dataset	Independent learner	Joint learner
Wikipedia	2119	1785
User Click	1572	927

Table 4. Advantage in terms of perplexity for joint learning.

5. Related Works & Discussions

In this paper we present LLA: Latent LSTM allocation. LLA leverages the powerful dynamic modeling capability of LSTM without blowing up the number of parameters while adding interpretability to the model. We achieve this by shifting from modeling the temporal dynamics at the observed word level to modeling the dynamics at a higher level of abstraction: topics. As the number of topics K is much smaller than the number of words V, it can act as a knob that can trade-off accuracy vs model size. In the extreme case, if we allow $K \rightarrow V$ in LLA then we recover LSTM. Furthermore, the topics provide an informative embedding that can reveal interesting temporal relationship as shown in Figure 6 and 7 – which is a novel contribution to the best of our knowledge.

Our work is related to various dynamic topic models, however, previous works like ddLDA (ddCRP in general) or hidden Markov topic models provide only limited modeling capabilities of the temporal dynamics. Specifically, they impose smoothness constraints: i.e. topic of a word is biased toward nearby topics. This constraint cannot learn a predictive action model, e.g. after "booking a flight" topic, the "booking a hotel" topic is likely to follow. Moreover, Internet users often wander between related activities, e.g "booking a hotel" topic will happen shortly after "booking a flight", but not necessarily immediately as the user might have been interrupted by something else. Thus we need a much richer temporal model such as an LSTM. Our quantitative results against ddLDA confirm this claim.

Another similar work would be lda2vec (Moody, 2016), where LDA is combined with local context in a fixed window size. This provides a sparse, interpretable model but lacks modeling the temporal aspect. The model cannot be used in a predictive-action setting. Also the sparsity is only in terms of per document parameters, whereas it suffers from huge dense of size vocabulary times embedding size.

Another relevant recent work is Sentence Level Recurrent Topic Model (SLRTM) (Tian et al., 2016). However, this model cannot capture long-range temporal dependencies, as the topic for each sentence is still decided using an exchangeable (non-sequential) Dirichlet-multinomial scheme similar to the LDA. It might be able to preserve local sentence structure or grammar, but that is particularly not very useful for the task of user modeling. Furthermore, as it contains RNN that operates on the entire vocabulary (not topic as in our case), the SLTRM has lots of parameters.

Finally our work is related to recent work in recurrent latent variable models (Chung et al., 2015; Gemici et al., 2017) where a recurrent model is endowed with latent variables to model variability in the input data. However, they mainly focused on continuous input space such as images and speech data which enables the use of variational autoencoder techniques to learn the model parameters. Whereas in this paper, we focus on discrete data that are not amenable to the standard variational autoencoder techniques and as such we developed an efficient SEM algorithm instead.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Ahmed, Amr and Xing, Eric. Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 219–230. SIAM, 2008.

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

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

Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

Blei, David and Frazier, Peter I. *Distance Dependent Chinese Restaurant Processes*. JMLR, 2011.

Cho, Sébastien Jean Kyunghyun, Memisevic, Roland, and Bengio, Yoshua. On using very large target vocabulary for neural machine translation. 2015.

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

Gemici, Mevlana, Hung, Chia-Chun, Santoro, Adam, Wayne, Greg, Mohamed, Shakir, Rezende, Danilo J, Amos, David, and Lillicrap, Timothy. Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*, 2017.

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

Hu, Diane J, Hall, Rob, and Attenberg, Josh. 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*, pp. 1640–1649. ACM, 2014.

Jozefowicz, Rafal, Vinyals, Oriol, Schuster, Mike, Shazeer, Noam, and Wu, Yonghui. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Karpathy, Andrej, Johnson, Justin, and Fei-Fei, Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Korpusik, Mandy, Sakaki, Shigeyuki, and Chen, Francine Chen Yan-Ying. Recurrent neural networks for customer purchase prediction on twitter. *CBRecSys 2016*, pp. 47, 2016.

Li, Aaron Q, Ahmed, Amr, Ravi, Sujith, and Smola, Alexander J. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 891–900. ACM, 2014.

Ling, Wang, Luís, Tiago, Marujo, Luís, Astudillo, Ramón Fernandez, Amir, Silvio, Dyer, Chris, Black, Alan W, and Trancoso, Isabel. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

Lipton, Zachary C, Berkowitz, John, and Elkan, Charles. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

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

Moody, Christopher E. Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv:1605.02019*, 2016.

Nielsen, Søren Feodor. The stochastic em algorithm: estimation and asymptotic results. *Bernoulli*, pp. 457–489, 2000.

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

Sundermeyer, Martin, Schlüter, Ralf, and Ney, Hermann. Lstm neural networks for language modeling. In *Interspeech*, pp. 194–197, 2012.

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

Tassarotti, Joseph and Steele Jr, Guy L. Efficient training of lda on a gpu by mean-for-mode estimation. In *32nd International Conference on Machine Learning ICML*, 2015.

Tian, Fei, Gao, Bin, and Liu, Tie-Yan. Sentence level recurrent topic model: Letting topics speak for themselves. *arXiv preprint arXiv:1604.02038*, 2016.

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

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

Zaheer, Manzil, Ahmed, Amr, Ravi, Sujith, and Smola, Alex. Fast sampling algorithms for sparse latent variable models. *arXiv preprint*, 2017.